instituto politécnico de leiria
escola superior de tecnologia e gestão

IPv6@IPLeiria
IPLeiria's Network IPv6 Migration

Roberto Emanuel Fernandes Leal
Tiago Miguel Santos de Almeida

July-2011

Work performed within the class of Final Project

IPv6@IPLeiria
IPLeiria's Network IPv6 Migration

Roberto Emanuel Fernandes Leal
Tiago Miguel Santos de Almeida

July-2011

Work performed within the class of Final Project, course of Computer Engineering
under the guidance of teacher Nuno Veiga

Acknowledgements

We would like to dedicate this work to our parents, brothers and sisters for all the support they gave us for this project.

A thank you and an acknowledgment and best wishes to both our colleagues, Carlos Silva and Bruno Silvestre, who worked parallel to us on their module of the project and who allowed us to mutually help each other.

We would like to give a very big thank you and special appreciation to our professor and mentor Prof. Nuno Miguel Afonso Veiga who by his teachings, demonstrated friendship, trust, encouragement and solidarity during the course of this project.

Special thanks to Eng. Adaíl Domingues da Silva de Oliveira, IPLeiria's IT Center Network Management and Security Team (UARS) engineer, for his enthusiasm and willingness to provide us with all the help we needed and for his readiness to assist us in solving existing problems and for providing us with the equipment necessary for the proper conclusion of the project.

(Tiago) I would like to give a very special thank you to my girlfriend for all the support and understanding she provided me with.

We thank all our friends and family who supported us, who gave us the strength and courage to complete this project.

Finally we would like to thank all those who directly or indirectly contributed to this project.

Abstract

IPv6 is the next step in the networking world. With the exhaustion of the IPv4 addressing range and the growth of the Internet user base the need to develop a newer and more flexible IP protocol has been arising for nearly twenty years and now the transition from IPv4 to IPv6 has begun. The idea behind a good IPv6 transition is to be able to keep the current IPv4 settings while, at the same time, implementing IPv6 into a network, allowing for an incremental growth of the new protocol while making sure IPv4 access to content and services remains unhindered.

In the development of this project we researched the current state of IPv6 transition technology, evaluated which solutions can be used within the IPLeiria's network and test those solutions. Along with these objectives this project also serves to define rules and addressing concepts used to create a structured addressing plan for the IPLeiria's network. An example addressing plan is given and alternatives are also defined as well as scalability and growth possibilities for the chosen addressing plan. Along with these studies this project also describes the implementation of a Dual Stacked SIP based VoIP server solution for IPv4 and IPv6 that allows for VoIP calls over IPv4 and IPv6.

Index

List of Figures

List of Tables

List of Acronyms

AH	Authentication Header
APIPA	Automatic Private Internet Protocol Addressing
ARP	Address Resolution Protocol
ARPANet	Advanced Research Projects Agency Network
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DHCPv4	Dynamic Host Configuration Protocol version 4
DMZ	Demilitarized Zone
DNS	Domain Name System
ESP	Encapsulation Security Payloads
ESTG	Escola Superior Tecnologia e Gestão de leiria
FCCN	Fundação para a Computação Cientifica Nacional
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
HETB	Hurricane Electric's Tunnel Broker
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
ICV	Integrity Check Value
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IM	Instant Messaging
IOS	Internetwork Operating System
IP	Internet Protocol
IPLeiria	Instituto Politécnico de Leiria
IPng	Internet Protocol next generation
IPSec	Internet Protocol Security Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IRC	Internet Relay Chat
ISAKMP	Internet Security Association and Key Management Protocol
ISATAP	Intra-Site Automatic Tunnel Addressing Protocol
ISP	Internet Service Provider
IT	Information Technology
LAN	Local Area Network
LCA	Laboratório de Comunicações Avançado

LIR	Local Internet Registries
MAC	Media Access Control Address
MTU	Maximum Transmission Until
NAT	Network Address Translation
NAT-PT	Network Address Port Translation + Protocol Translation
ND	Neighbor Discovery Protocol
NLA ID	Next Level Aggregation Identifier
OS	Operating System
OSPF	Open Shortest Path First
OSPFv3	Open Shortest Path First version 3
PBX	Private Branch Exchange
PC	Personal Computer
PDA	Personal Digital Assistants
POTS	Plain Old Telephone Service
QoS	Quality of Service
RADvD	Router Advertisement Daemon
RES	Revered for future used
RFC	Request For Comments
RTP	Real-Time Transport Protocol
SIP	Session Initiation Protocol
SLA ID	Site Level Aggregation Identifier
TCP	Transmission Control Protocol
TLA ID	Top Level Aggregation Identifier
ToS	Type of Service
TTL	Time To Live
UARS	Network Management and Security Team
UDP	User Datagram Protocol
VLAN	Virtual Local Area Networks
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WAN	Wide Area Network

Chapter 1 Introduction

With the exhaustion of the IPv4 addressing range IPv6 transition becomes more and more a priority. The percentage of world population with access to the Internet is growing at a daily basis and IPv4 is unable to keep up. IPv6 transition is becoming a necessity however its adoption won't come as a big cataclysmic event that instantly migrates all equipment from IPv4 to IPv6. Instead a structured and defined transition plan must be studied and developed.

1.1 Motivation

IPLeiria not wanting to remain behind on the IPv6 transition field wants to prepare itself for the coming future. As such the need arose to create an IPv6 test bed where some basic services would be configured and tested so begins the process of studying the implementation of a structured transition plan for the IPLeiria's network.

The strongest motivation we have had along the development cycle of this project was given to us at the very start. We were told we would be the first educational Institute in the whole country to study and prepare an IPv6 transition plan. This got both of us very excited.

The fact that IPv6 is a new technology was also very motivating, we would be studying and implementing network solutions that are now starting to be used which allows us to gain a broader and wider view on the whole IPv6 theme, something that will become very interesting in the near future as more and more institutions and networks start their own transition process.

World IPv6 day was also something that caught our attention and spiked our interest. The chance to publicly allow access to the work that has been done, by us and other students to the whole world was indeed a great and interesting idea. Participation in the world IPv6 day brings notoriety and attention not only to the work that we are doing but also to the school as a whole. The fact that IPLeiria was

the only Portuguese education institution to actually take part in the world IPv6 day also served to boast our motivation even further.

1.2 Objectives

With this project and report we were intended to research, test and present a solution for the transition from IPv4 to IPv6 of the IPLeiria's network. With the development of this report we were tasked with the proposal of a solution for IPv6 transition incorporating various tunneling mechanisms for IPv6 encapsulated in IPv4 for PCs and routing equipment as well as a fully structured IPv6 addressing plan based on a use type and location paradigm. Following the request of the IPLeiria's IT center UARS we were also to setup, test and document a solution for a dual stacked SIP VoIP server that allows both IPv4 and IPv6 only clients to connect as well as make VoIP calls between them. Along with this module a second module was also prepared by our colleagues which, amongst other things, sets up some basic network services like address auto-configuration (via Router Advertisement and DHCPv6), DNS and IPv6 Web Hosting.

1.3 Planning

The first task that was appointed to us was to study the transition solutions available out there to figure out what had changed in between 2007 and 2011 which was the time of the previous IPv6 study developed by IPLeiria students. We were to evaluate the changes and updates that IPv6 and its transition plans had received over the years. After this first task tests concerning IPv4 tunneling were to be conducted in order to verify, test and evaluate the transition mechanisms at work.

As a second task we were to study a solution for a structured IPv6 addressing plan. Rules and concepts were to be defined in order to create a possible addressing plan to implement within the IPLeiria's network. Alternative addressing plan options were welcomed along with an easy way to scale the possible solution with the future growth of the network.

Thirdly as the IPLeiria depends on a VoIP solution for internal telephony we were tasked with the research and implementation of an IPv6 SIP VoIP server. After the initial setup of the server test calls were to be made in order to evaluate the compatibility of the existing equipment namely some SIP Phone models made available by the IPLeiria's IT center UARS.

Chapter 2 Reasons for IPv6

Internet Protocol version 4 or IPv4 as its broadly known was developed by Vint Cerf, who is known as one of the fathers of the internet, and was first specified in the RFC 791 in January of 1981, this version of the protocol was robust and of easy implementation [1]. Since it uses 32 bit addresses it allows for a total of up to roughly 4.3 billion addresses.

Initially, before the Internet boom, the computer network was called ARPAnet and was conceptualized for institutions such as universities and the military, it was thought of as a means to share information within a very private and trust based network. In this context, IPv4 was a more than suitable protocol to use and implement but with the passing of the years these requirements have changed [2].

During its conception, some important notions were not taken into account. In February 2011 IANA gave out the last block of available IPv4 addresses leading to the extinction of the free IPv4 addressing space something that was already foreseeable since the 80s.

The exponential growth of the internet user base mainly in some oriental countries like China and India, the existence of a lot of always-on equipment such as broadband connections in a large amount of homes and offices allied to the huge and sudden growth in mobile consumer devices such as smart phones and PDAs reveal both a need to increase the available addressing space as well as the requirement for easier auto configuration mechanisms. All of these aspects contributed to the starvation of the aforementioned addressing space and led to its exhaustion.

Also, adding to the addressing space exhaustion predicament, there are issues concerning QoS. Although IPv4 supports QoS, specified in the ToS field of the IPv4 header this kind of QoS implementation has limited functionality due to the fact that encrypted traffic cannot be identified and QoS is applied per packet. [3].

2.1 Major changes in IPv6

IPv6 was created in 1995 by the IETF (Internet Engineering Task Force) under the name IPng (Internet Protocol next generation) and was first described in the RFC 2460 in December 1998 [4]. IPv6 uses 128 bit addresses which allows for up to around 3.4x10^38 addresses, a number hugely superior to the one provided by IPv4 which allows networks to support a larger number of hierarchical levels.

During the development of IPv6 a lot of lessons were taken in from IPv4 and some of the mistakes and less important features were either removed or changed, for example the IPv6 header has a static length and less fields than the IPv4 header. The checksum is gone and extra options are now implemented through the use of extension headers. This allows for easy implementation of new features and for a simplified evolution of the protocol. IPv6 also possesses QoS features applied to the stream and not just individual packets. This flow branding capability lets the network equipment treat a whole flow of data applying QoS to it instead of doing it packet to packet like previously happened [5].

But not everything about IPv6 is a positive change, older equipment is not ready to "understand and speak" the new protocol. This means that transition towards IPv6 has to be phased and cannot be rushed. There is a need to implement transition mechanisms that allow for the new protocol to be used while keeping the compatibility with the older protocol intact.

Due to this very specific need a transition solution needs to be studied and tested before it is implemented and this is the challenge we were proposed to take part in with the development of this project.

Table 1 depicts a few of the major changes from IPv4 to IPv6.

Table 1 - IPv4 vs. IPv6 [6, pp. 64-66]

Implemented Changes	**IPv4**	**IPv6**
Source and Destination addresses	32 bits (4bytes)	128 bits (16bytes)
IPSec	Optional	Mandatory (RFC 2401)
QoS	Is not able to identify QoS packet flows, therefore relying on QoS per packet.	IPv6 utilizes the flow label field in its header to identify a QoS packet flow.
Fragmentation	Fragmentation is done by	Fragmentation is done only

	the source node as well as the path routers which give way to performance issues.	by the source node because it is able to determine the smallest MTU in the path.
Header Checksum	Header includes checksum	Header does not include checksum
Options	Header includes options	All additional information is stored in the optional extension headers
Address Resolution	The ARP mechanism utilizes Broadcast ARP request frames to resolve IPv4 addresses into network layer addresses (MAC)	ARP Request frames are replaced by Multicast Neighbor Solicitation messages
ICMP auto configuration	Uses the ICMP router discovery protocol.	ICMP Router discovery is now done through ICMPv6 router Solicitation and Router Advertisement messages
Broadcast	Uses broadcast addresses	Broadcast addresses are deprecated. Instead a link local address range multicast address is used.
Interface configuration	Manual or DHCP configuration.	Auto configuration does not require the use of DHCP.
DNS	Use of address data bases (DNS) to map machine names to IPv4 addresses.	Uses AAAA DNS registries to map machine names to IPv6 addresses.
DNS domain	Uses the DNS domain pointer IN-ADDR.ARPA to map IPv4 addresses to the respective machine names.	Uses the DNS domain pointer IP6.INT to map IPv4 addresses to the respective machine names.
Packet size	Supports a packet size of 570 bytes (possibly fragmented)	Supports a packet size of 1280 bytes (without fragmentation)

2.2 Comparison between the IPv4 and the IPv6 header

The IPv6 header is an evolution of the older IPv4 header. Some fields are removed and new ones emerge, the IPv6 header consists of the basic header and the expansion headers which define other options. The only fields that were kept in the base header are Version, Source Address and Destination Address obviously because the equipment needs to be able to identify the protocol version as well as the source and destination addresses.

Figure 1 shows a comparison between the IPv4 and the IPv6 header. Notice how this time around the header has been simplified.

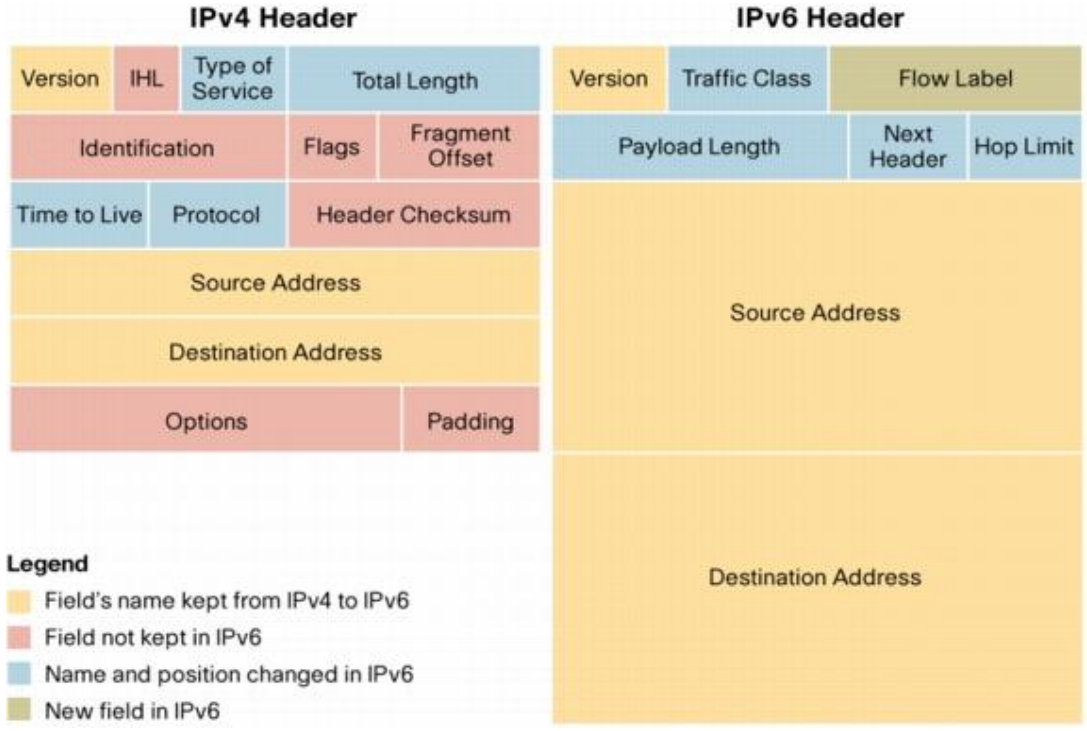


Figure 1 - IPv4 vs. IPv6 headers [7]

Regarding the IPv6 header structure it is easy to notice how much it has been simplified, whereas the IPv4 header consists of 14 fields the IPv6 header consists of only 8. Some of the fields were kept such as the version field, which identifies which version of the IP protocol is used as well as the source and destination addresses which, despite possessing a much larger size, serve the exact same purpose as they did in the IPv4 header. The Traffic class and Flow Label fields are used for QoS in purposes. The Payload length shows the size of the data field. Next Header indicates the type of the following extension header. Hop limit has the same objective and follows the same principle as the TTL field in the IPv4 header.

2.3 Mobile IP, QoS and IPSec in IPv6

IP Mobility, Quality of Service and IPsec are three of the major features of IPv6. Being a technology focused on mobility and always-on connectivity, some concerns regarding QoS and more importantly, security are raised. This sub-chapter aims to introduce some of the features and concepts relevant to these issues as well as explain how these technologies are implemented.

2.3.1 Mobile IP

Mobile IPv6 aims to enable mobile terminal roaming in IPv6 networks. It is specified in the RFC 6275 [8]. This technology specifies the ability to change the point of attachment in the IPv6 network without changing the node's IPv6 address. Figure 2 depicts the operating mechanism in which each mobile terminal (node) has two or more IP addresses, one static, with the name of "home address", and one or more variable addresses called "care-of address(es)", which are obtained and used when the mobile node is outside the range of the home agent. In this case the mobile host connects to the foreign agent which is connected to the home agent and through which connection the packets destined to the mobile hosts home address are rerouted from the Home agent to the connected foreign agent (connected to the care-of address) and therefore to the mobile node [9].

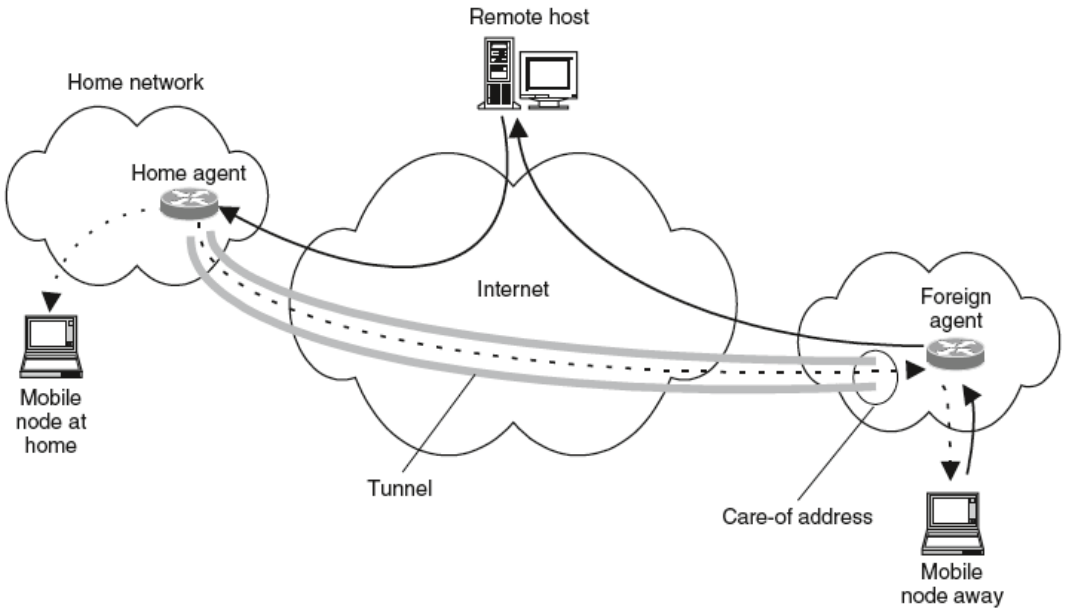


Figure 2 - Operation of Mobile IPv6 [10]

2.3.2 QoS in IPv6

Quality of service is everyday a more important and emerging feature of modern networks. With this idea in mind IPv6 was developed with a set of QoS features that are much more reliable and functional than the ones implemented in IPv4. QoS in IPv6 is able to determine which packets of data belong to time sensitive applications such as VoIP calls or Video Streaming and which ones do not [11].

The IPv6 header was defined with advanced QoS features in mind. These features use both the traffic class and flow label fields of the IPv6 header. The traffic class field is a set of 8bits used to differentiate packets between different classes or priorities and it works much like the ToS header field did in IPv4. The Flow Label field is composed of a set of 20bits that differentiates packet streams. This differentiation is made by the source and is not altered in the network allowing packets belonging to the same stream to be awarded the same type of service without the need for a packet-by-packet approach, therefore increasing performance and reducing computing and routing time [11].

In addition, fragmentation issues have also been addressed. No longer is fragmentation applied by the routing equipment along the path but now it is made by the source host. It probes the different links along the path towards the destination address, determines the smallest MTU along that path and sends the packets sized accordingly. This way there is no time lost with fragmenting packets while they travel through the network thereby increasing performance [12].

2.3.3 IPSec in IPv6

IPSec, Internet Protocol Security, is defined in the RFC 6071, as IPv4 was not designed for security IPSec was a mechanism developed after the adoption of IPv4 to implement various levels of network security [13]. It implements encryption and authentication at the network layer and ensures point-to-point security in packets by ensuring integrity of information, confidentiality and authentication. The operation of IPSec is similar in both IPv4 and IPv6 and while it is not required in IPv4 its implementation in IPv6 is mandatory and all equipment supporting IPv6 and therefore all equipment supporting IPv6 must support IPSec. Along with the fact that IPv6 aims to provide public connectivity to every node connected to the network, therefore dismissing the use of private addresses this mandatory nature of IPSec makes NAT obsolete [14]. In addition to the loss of need for NAT there is also a stronger reason behind the fact of its abandonment, the fact that IPSec will not work with NAT since it breaks end-to-end connectivity (RFC 2993) because IPSec's authentication check covers the TCP/UDP checksum (which in turn covers the IP address) [15]. When a NAT changes the IP address, the checksum calculation will fail, and therefore authentication is guaranteed to fail because only the end points have access to the keys and therefore are the only ones that can manipulate the headers and re-calculate the checksum.

IPSec uses two extension headers, AH, Authentication Header is defined in the RFC 4302 [16], and ESP, Encapsulation Security Payload, defined in RFC 4303 [17]. IPSec uses a standard key management protocol, ISAKMP, defined in the RFC 4306 and RFC 5996, which provides a framework for authentication and key exchange but does not define them [18], [19]. It is designed to support many different "kinds" of key exchanges (different set of rules and options). IPSec also uses the Internet Key Exchange protocol (IKE) which is defined in the RFC 2409 and is used to define a set of rules and options for the use of the ISAKMP protocol. It implements a subset of the OAKLEY Key Determination for Protocol necessary to satisfy its goals [20].

The IPSec operation is similar in both IPv4 and IPv6 and as previously stated its implementation in IPv6 compatible equipment is mandatory. IPSec possesses two modes of operation, transport mode (host-to-host) where the payload is encapsulated and the header is left intact, the end-host, or destination address, then de-capsulates the packet. The other mode of operation is tunnel mode (Gateway-to-Gateway or Gateway-to-host) which works by encapsulating the entire IP packet (with a new header). The host (or gateway) specified in the new IP header de-capsulates the packet.

In IPv6 IPsec is implemented using the AH authentication header and the ESP extension header [21].

2.4 Synthesis

This chapter summarizes the history and description of IPv4 and IPv6. In this chapter we compared some of the functionalities that both these versions of the IP protocol provide as well the changes suffered by the IPv6 header when compared to the older IPv4 header. We also approached some important IPv6 specifications such as IP Mobility, QoS and IPSec, specification that, in spite of not being new technologies, since they are also defined for IPv4, are implemented from scratch and were taken into account when creating the protocol instead of being patched in later like it happened with IPv4.

Chapter 3 IPv6 Addressing

An IPv6 address is composed of 128 bits displayed in an hexadecimal format with each 4 hexadecimal digits separated by a colon (:) contrary to an IPv4 address which is composed of 32 bits displayed in a decimal format with each 3 digits separated by a dot (.).

There are 3 types of IPv6 addresses which are Unicast, Multicast and Anycast. Both Unicast and Multicast were already present in IPv4 and their implementation and aim remain the same; however Anycast addresses are a new addition to the IP addressing space. Broadcast addresses have also suffered a change and are this time not present at all in the new version of the IP protocol being totally deprecated this time around.

Both Unicast and Anycast addresses in IPv6 have the following scopes (multicast addresses have their own scope built into them):

- **Link-local:** The local link directly connected to the equipment (nodes in the same sub-net)
- **Site-local:** the site, or organization, in which the nodes are, located (private site addressing) this scope of addresses has been deprecated and should not be used.
- **Global:** Global addresses. Connectivity available from everywhere (IPv6 Internet addresses)

3.1 Unicast Addresses

Defined in RFC 3587 [22]. There are various types of IPv6 unicast addresses, the major ones being.

3.1.3 Unicast Global addresses

IPv6 unicast global addresses work very much alike IPv4 public addresses in that they aim to always be reachable through and from the internet. Also known as aggregatable global unicast addresses, global addresses are globally routable. Their structure creates the three-level topology, consisting of public topology (48 bits), site topology (16 bits) and Interface ID (64 bits). Figure 3 specifies said structure:

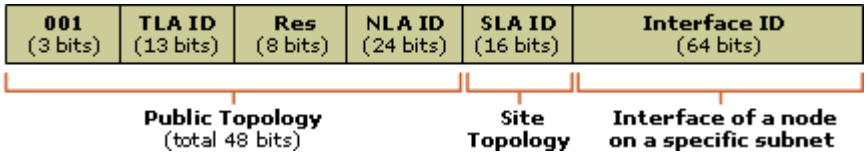


Figure 3 - IPv6 unicast global address fields [23]

Table 2 enumerates the fields that compose an IPv6 unicast global address and describes them.

Table 2 - IPv6 unicast global address fields [24]

Field	Description
001 (3bits)	Identifies the address as being an IPv6 global unicast address.
Top Level Aggregation Identifier (TLA ID) (13bits)	Identifies the highest level in the routing hierarchy. TLA IDs are administered by IANA. They are allocated by IANA to local Internet registries, which proceed to allocate a specific TLA ID to a global ISP.
Res (8bits)	Reserved for future use (to expand either the TLA ID or the NLA ID if/when needed)
Next Level Aggregation Identifier (NLA ID) (24bits)	Identifies a specific customer site
Site Level Aggregation Identifier (SLA ID) (16bits)	Defines an Intra-Site subnet ID and enables as many as 65,536 (2^16) subnets within that same individual organization's site. Assigned within the site, the ISP cannot change the SLA ID part of the address since.
Interface ID (64bits)	Identifies the interface of a node (host/router/equipment) on a specific subnet.

3.1.4 Unicast Site-local addresse

IPv6 unicast site-local addresses work similarly to IPv4 private addresses and aim to achieve the same basic principles in that they both are routable within a specific addressing space and will not be forwarded outside of those same subnets. In other words, the scope of a site-local address is the internal network of an organization's site. (Both global and site-local addresses can be used in a network.) The prefix for site-local addresses is FEC0::/48.

Figure 4 shows the structure of a site-local address.

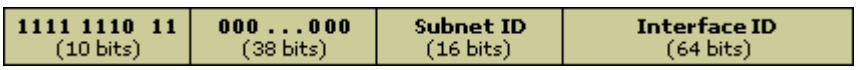


Figure 4 - IPv6 unicast site-local address scheme [23]

The initial 48 bits are fixed to describe a unicast site-local address and are followed by a 16-bit **Subnet ID** field, which provides as many as 65,536 (2^16) subnets in a flat subnet structure. It is also possible to subdivide the high-order bits of the **Subnet ID** field to create a hierarchical routing infrastructure. The last field is a 64-bit **Interface ID** field that identifies the interface of a node on a specific subnet.

As an additional note it is important to take notice that Unicast site-local addresses have been deprecated and their use has been abandoned, these addresses should not be used. [25].

3.1.5 Link-local addresses

IPv6 unicast link-local addresses are similar to IPv4 APIPA addresses used by computers running Microsoft Windows. The prefix for link-local addresses is FE80::/64. Nodes on the same subnet (or link) use these addresses, which are configured automatically, to communicate between each other. Neighbor Discovery provides address resolution. The following illustration shows the structure of a link-local address.

3.1.6 Unicast Unspecified address

The IPv6 unicast unspecified address is equivalent to the IPv4 unspecified address of 0.0.0.0. The IPv6 unspecified address is 0:0:0:0:0:0:0:0: or a double colon (::).

3.1.7 Unicast Loopback address

The IPv6 unicast loopback address is equivalent to the IPv4 loopback address, 127.0.0.1 also known as local-host. The IPv6 loopback address is 0:0:0:0:0:0:0:1, or ::1.

3.1.8 Unicast 6to4 addresses

IPv6 uses 6to4 addresses to communicate between two IPv6 nodes over an IPv4 network. A 6to4 address combines the prefix 2002::/16 with the 32 bits of the IPv4 address of the node to create a 48-bit prefix — 2002:*WWXX*:*YYZZ*::/48, where *WWXX*:*YYZZ* is the colon-hexadecimal representation of *w.x.y.z*, an IPv4 address. Therefore, the IPv4 address 157.60.91.123 translates into a 6to4 address prefix of 2002:9D3C:5B7B::/48. Along with ISATAP, 6to4 is a very flexible way to deploy IPv6 connectivity over an IPv4 network. Figure 5 illustration shows the structure of a 6to4 address.

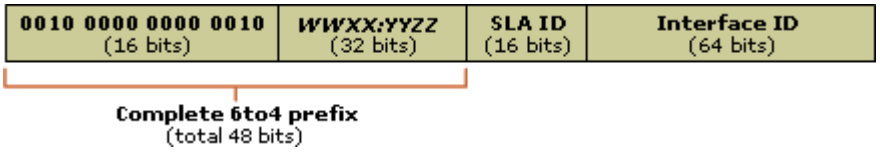


Figure 5 - IPv6 unicast 6to4 address scheme [23]

This method is called tunneling and is later on in this report delved into in more detail.

3.1.9 Unicast ISATAP address

IPv6 can use ISATAP addresses to communicate between two IPv6 nodes over an IPv4 intranet. An ISATAP address combines a 64-bit unicast link-local, site-local, or global prefix (a global prefix might be a 6 to 4 prefix) with a 64-bit suffix constructed using the ISATAP identifier 0000:5EFE (a modified eui-64 identifier), followed by the IPv4 address assigned to an interface of the host. The prefix is known as the *subnet prefix*. ISATAP addresses are used within a specific site and exist to provide IPv6 over IPv4 automatic tunnels within that same site. Along with 6to4, ISATAP is a very flexible way to deploy IPv6 connectivity over an IPv4 network. Figure 6 shows the structure of an ISATAP address [23].

Figure 6 - IPv6 unicast ISATAP address scheme [23]

3.2 Multicast IPv6 Addresses

IPv6 multicast addresses work similarly to IPv4 multicast addresses. Packets forwarded to a multicast address are delivered to all interfaces that the address identifies.

Figure 7 shows the structure of an IPv6 multicast address.

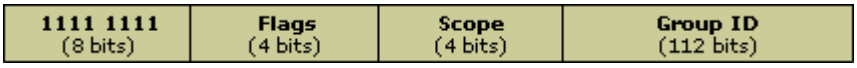


Figure 7 - IPv6 multicast address scheme [23]

Table 3 enumerates the fields that compose an IPv6 multicast global address and describes them.

Table 3 - Multicast address fields [23]

Field	Description
1111 1111	Identifies the address as being an IPv6 multicast address.
Flags	This field originally included only one flag, the T (Transient) Flag to indicate a transient address but it has evolved and the RFC 3306 has added the P (Prefix) flag used for allowing part of the group address to include the source network's Unicast prefix, which creates a globally unique Group Address. [26]
Scope	Indicates the scope of the multicast traffic, such as link-local, site-local, organization-local, or global scope.
Group ID	Identifies the Multicast Group

3.2.1 Multicast Solicited node address

The IPv6 multicast solicited node address is used for efficient address resolution and it works to achieve the same end as the IPv4 ARP Request but in a much less invasive manner. In IPv4, the ARP Request frame is sent to the MAC-level broadcast, which disturbs all nodes on the network segment. However, in IPv6 the multicast solicited node address combines the prefix FF02::1:FF00:0/104 with the last 24 bits of the IPv6 address being resolved. IPv6 uses the solicited node multicast address for the Neighbor Solicitation message (the equivalent in IPv6 to the ARP Request frame) that resolves an IPv6 address to its link-layer address while disturbing as few nodes as possible during the process of address resolution.

3.3 Anycast IPv6 Addresses

Anycast IPv6 addresses are similar to the ones used in IPv4 but are much more efficient. These addresses are mainly used by large ISPs. Anycast addresses use the unicast address space but function differently from other unicast addresses. IPv6 uses anycast addresses to identify multiple interfaces. IPv6 delivers packets that are addressed to an anycast address to the nearest interface that the anycast address identifies. In contrast to a multicast address, where delivery is done to all the identified addresses in a one too many approach, an anycast address delivery is from one to one-of-many.

3.4 Synthesis

In this chapter we specify the different IPv6 addresses that exist and class them within the three different IPv6 address types (Unicast, Multicast and Anycast) and IPv6 address scopes (Link local, Site local, Global). Examples of different types of these addresses are given as well as an explanation for each example.

Chapter 4 Transition Mechanisms

Transition mechanisms are called this way because their aim is to facilitate the exact process it describes, to ease transition between the wide adoption of one protocol version over the other, in this case the adoption of IPv6 and the paced abandonment of IPv4. Transition Mechanisms are separated into 3 categories: Dual-Stack, Tunneling and Translation Mechanisms.

4.1 Dual Stack Mechanisms

This type of mechanism is defined in the RFC 4213 [27]. This kind of mechanism consists in the fact that one single interface has both IPv4 and IPv6 versions installed and is able to comprehend both of them effectively being able to communicate natively in either one.

As an easy to install, implement and manage mechanism, a dual stack solution is a great way to maintain IPv4 connectivity and support while at the same time taking advantage of the newer protocol version and the benefits it brings.

Derived from having to run both protocols at once and needing to distinguish between packets a dual stack solution will always require two separate routing tables and higher resource consumption such as more CPU and memory requirements [28].

4.2 Tunnel Mechanisms

A tunnel is a virtual interface with two or more end points. These end points are required to have both an IPv4 and an IPv6 address. They work much like a loopback address, in that there is no equivalent physical interface associated with the logical interface but are notoriously different in that there are always two or more interfaces associated with them. These virtual interfaces are called end-points and make up the tunnel edges, the points where the packets enter and leave the tunnel are tasked with the encapsulation and de-capsulation of those packets. Also worthy of noting is the fact that both end points of the tunnel need to be running a dual stack mechanism due to the fact that they must be able to understand and process IPv6 packets and forward them through the IPv4 network meaning it is mandatory to run both versions of the IP protocol.

Tunnel Mechanisms are subdivided into various types which are Manual, Semi-Automatic and automatic [28].

4.3 Manual Tunnels

Manual tunnels are used to create a point-to-point IPv6 virtual link. This kind of tunnel is mostly used in a router-to-router connection and its usual purpose is to connect two separate IPv6 islands together.

Although these tunnels can be quite useful the fact remains that they do suffer from a hindering lack of scalability. It is relatively simple to manually configure one tunnel but if we are talking about a huge network with thousands of routers and a lot of tunnels need to be configured it can become a very tiresome and exhausting endeavor.

This kind of tunnels can be further divided into Manual tunnels and GRE tunnels [28].

4.3.1 Manual Tunnels

First mentioned in the RFC 2893 [29] having later been replaced by the RFC 4213 [27]. This kind of tunnel allows IPv6 packets to travel and be forwarded in the IPv4 network. The tunnels destination end-point address is configured in the tunnels source end-point interface.

When forwarded by this kind of tunnel the IPv6 packet only sees it as one hop because the IPv4 links through which the IPv6 packet is forwarded through are invisible to the IPv6 packet which is only able to take notice of the single hop from one end-point of the tunnel to the other no matter how many hops the encapsulating IPv4 packet has to go through.

Manual tunnels are easy to implement and widely available in the majority of existing platforms. Due to the fact that they must be manually configured their scalability suffers greatly, the fact that they possess problems related with delay and

latency, excessive CPU usage and the existence of a single point of failure also hinders them and adds to their disadvantages list [28].

4.3.2 GRE Tunnels

Defined in the RFC 2473 [30] this type tunnel shares the same principles with the Manual tunnels but with the difference that the packets are encapsulated in a specific header (instead of inside an IPv4 packet) inside virtual point-to-point links (tunnels) over an IP network. It was developed by Cisco and can encapsulate a wide variety of network layer protocols.

This makes it possible to have IPv4 and IPv6 networks both running over the same single backbone. These tunnels can connect discontinued networks and allow for VPNs and WANs. But they also possess downsides like if one of the end-points of the tunnel is down the other end-point is not notified and will keep forwarding packets through the tunnel resulting in a total loss of data [28].

4.4 Semi-Automatic Tunnels

Also called Tunnel Brokers were first specified in the RFC 3053 [31] and were developed with the help of an IETF group. This kind of tunnel uses a dedicated server that automatically configures the tunnel for the user.

This kind of tunnels are able to better manage the users accessibility to the IPv6 network because it becomes a lot easier for single, isolated hosts to be able to connect to the IPv6 network through a tunnel over the IPv4 network without having to manually set up a tunnel themselves.

This kind of tunnels need to give special attention to security issues and NAT problems due to the nature of the auto configuration tunnel and the security issues that are raised by giving that kind of administrative privilege to an outside source [28].

4.5 Automatic Tunnels

These kinds of tunnels were developed to allow coexistence between IPv4 and IPv6 networks with relative lack of node configuration. In these kinds of mechanisms the end point's IPv6 addresses are mapped using their configured IPv4 address and are attributed automatically without the need for manual configuration.

There are several kinds of automatic tunnels:

- Teredo
- IPv4 Compatible IPv6
- 6to4
- 6over4
- ISATAP

4.5.1 Teredo Tunnels

This type of tunnel was developed by Microsoft and is described in the RFC 4380 [32] . One of its characteristics is allowing IPv6 packets to get through NAT which means they enable private IPv4 networks that are behind a NAT equipment to be able to connect to an outside IPv6 network.

It works by encapsulating the IPv6 packets in IPv4 UDP datagram's and forwarding them through a tunnel to an IPv4 Teredo server. These servers are stateless and their job is to manage the traffic between Teredo clients. In order to communicate with the native IPv6 internet, Teredo relays are used between Teredo clients and Teredo servers. These relays act as IPv6 routers and allow the clients connected to the Teredo server access to the native IPv6 internet.

Although this kind of tunnels work with private IPv4 addresses, the fact that they possess a significant amount of configuration requirements as well as a high demand for administrative work and constitute possible security problems have contributed to the gradual abandonment of this technology. This is mainly due to the fact that a private IPv4 addressed machine behind a NAT equipment becomes reachable from the internet due to the fact that the Teredo server routes to it a public IPv6 address and in doing so Teredo potentially exposes, any IPv6 enabled application with an open port, to the outside [28].

4.5.2 IPv4 Compatible IPv6 Tunnels

Fist described in the RFC 2893 [29] and later on replaced by the RFC 4213 [27]. This kind of tunnels' main characteristic is encapsulating IPv6 packets in IPv4 packets and routing them through the IPv4 network without needing to pre-configure the tunnel end point on the destination side.

Being easy to implement without the need for pre-configurations and being able to work with IPv4 automatic addressing mechanisms as well as manual IPv4 address configurations were all good reasons to develop this technology. However a global IPv4 address is required to use these tunnels and it isn't able to route IPv4 packets to broadcast, multicast and loopback addresses. Due to these facts and the existence of alternatives this kind of tunnels has since been deprecated and abandoned [28].

4.5.3 6to4 Tunnels

6to4 tunneling was first defined in the RFC 3056 [33] as a stateless tunneling mechanism. This means that it treats each packet of data individually from the rest. The IPv6 packets are automatically encapsulated in IPv4 packets and forwarded from one end of the tunnel to the other through the IPv4 network in a way that is perceived by the end user as being a straightforward IPv6 connection. A 6to4 tunnel possesses a specific reserved prefix of 2002::/16 for its addressing purposes having said prefix been assigned by IANA.

The IPv4 dot decimal address format of the interface where the tunnel is configured is translated into its colon hexadecimal IPv6 address equivalent in which XXX.YYY.WWW.ZZZ translates into XXYY:WWZZ following the process described henceforth:

XXX in doted decimal format is converted into its hexadecimal equivalent along with YYY, WWW and ZZZ. In this example 192.168.1.2 translates into C0A8:12 because 192 in hexadecimal is C0, 168 in hexadecimal is A8, 1 in hexadecimal is 1 and 2 in hexadecimal is 2.

Therefore by adding the 6to4 prefix before the address translation we get the full 6to4 tunnel prefix for this end of the tunnel: 2002:C0A8:12::/48

Due to the fact that this is a stateless tunnel technology the routing delay in both ends of the tunnel can be a hassle, the IPv6 packet needs to be encapsulated when it enters the tunnel and de-capsulated when it exits it and having to repeat this process for every packet in a stream of data individually can considerably hinder network performance [28].

4.5.4 6over4 tunnels

First defined in the RFC 2529 [34] . The aim of this type of tunnels is to allow isolated IPv6 hosts to get link local connectivity with other hosts through the IPv4 network without the need to configure IPv4 connectivity. This is achieved by using the IPv4 multicast domain as the virtual local link for the IPv6 connectivity [28].

This kind of tunnels has already been abandoned [35].

4.5.5 ISATAP Tunnels

Defined in the RFC 4214 [36] and later specified in the RFC 5214 [37]. These tunnels' characteristic is that they are able to combine dual-stack and tunneling mechanisms.

ISATAP views the IPv4 layer as a link layer for IPv6. It works by placing the IPv4 gateway address in the ISATAP interface ID. ISATAP is usually used from within a site to acquire IPv6 connectivity from a host to the edge/border router which is in turn responsible for the IPv6 connectivity to the outside IPv6 network [28].

4.6 Translation

Also known as NAT-PT (Network Address Translation and Protocol Translation) and is defined in the RFC 4966 [38]. This type of mechanism enables IPv4 only networks and nodes to communicate with IPv6 only networks and nodes and vice versa. The idea behind this mechanism is analogous to the one behind NAT but instead of translating private IPv4 addresses to public ones this mechanism translates IPv4 addresses into IPv6 ones and vice versa allowing hosts with different IP versions to communicate with each other.

This mechanism should, however, only be used as a last resource due to the fact that it does not allow the use of advanced IPv6 resources [28].

4.7 Security Issues brought up by the IPv6 transition

Transition mechanisms are indeed helpful and aid us in traversing the gap between IPv4 and IPv6, helping to bring them both closer together and allowing for a smooth and bump free transition process. However they do bring up several security concerns and issues. We will depict some of these issues in order to raise the network user / administrator awareness so that they may be monitored and, if needed, action can be taken to lower or totally negate their repercussions.

Some transition Mechanisms come automatically enabled in most OSs and as such they might even be used or exploited without the consent of the user. These mechanisms can be leveraged or exploited to bypass firewalls and/or circumvent Network Intrusion Detection Systems. For example an IPv6 encapsulated in IPv4 tunnel might be used to bypass IPv4 firewall policies.

Other mechanisms, like Teredo, were designed specifically to bypass NAT, which makes up a basic protection solution by only allowing instances of communication that have been initiated from the internal network. By circumventing NAT these mechanisms become accessible and routable from the internet which consists of a big security concern.

Another issue that needs to be mentioned is Firewall policies. These are applied to IPv4 or native IPv6 but might not be fully compatible with some transition mechanisms. These policies function by analyzing the contents of the header fields

(TCP port numbers, IP protocol header field, etc) however the same firewall might be unable to filter packets with the same level of detail if transition mechanisms are employed because the information to be analyzed by the policies might be encapsulated in the data field of the analyzed packet.

But out of all transition mechanisms and the issues related to security that they bring up the most threatening one and the one that raises more concern is the automatic IPv6 tunnels. This is due to the fact that they might be used without prior consent of the user or network administrator. These might be very hard to police and monitor unless specific measures are taken like, for example, disabling the support for those kinds of mechanisms that comes automatically enabled in most operating systems.

Concerning Network Intrusion detection systems, these might be able to detect an IPv4 attack pattern but fail to detect the equivalent IPv6 attack pattern.

Firewalls might block certain incoming IPv4 connections from the internet but inadvertently allow IPv6 connections from the internet to hosts located inside the local network, effectively bypassing security restrictions installed by the network manager [39].

4.7.1 What to do and how to cope with these situations?

A possible solution is the default-deny approach to the policies. Only allow IPv6 traffic as a result of an explicit decision and every other traffic is blocked. However it is needed to point out that IPv6 may still be used from within the local network as a means of attack as any traffic located behind the firewall and within the "secured" zone therefore isn't exposed to the firewall filters. As such, an attacker with access to the local network might still be able to take advantage of the native IPv6 support of the network's local hosts to perform exploits and attacks against those hosts causing them to make use of their native IPv6 connectivity either unwillingly or unknowingly and possibly remaining undetected by security devices with no IPv6 support.

A way to counteract this is by implementing IPv6 traffic monitoring mechanisms in the local network which will raise the awareness of the IPv6 traffic traveling inside the network and allow for the network administrator to take notice if some out of the ordinary activity takes place.

As well, disabling IPv6 support in systems that are not expected to use it is also a good principle and the same goes for disabling support for the transition mechanisms that come pre enabled with a fresh install of most operating systems so that these may not be exploited.

Tunnel transition mechanisms are rather easy to identify and treat because most of them work by encapsulating an IPv6 packet inside an IPv4 packet and setting the protocol field of the IPv4 packet to 41 (IPv6). Still, special attention and care needs to be taken into consideration because some transition mechanisms use specific

methods to work. An example of this is Teredo tunnels where clients use UDP port 3544 to communicate with Teredo servers. By blocking all outgoing connections on UDP destination port 3544 we might be getting what are called a “false positive” blocks, other applications using UDP port 3544 and setting a destination that isn’t a TEREDO server are getting blocked when they should actually be allowed to get through. This can be changed by only blocking outgoing UDP port 3544 connections with the destination address set to known Teredo servers. It is up to the network administrator to specify in the firewall policy which IP addresses should be blocked. The exact same situation happens when dealing with Tunnel Brokers with Tunnel Setup Protocol (TSP) and the TCP/UDP port 3653. [39]

4.8 Conclusion

The IPLeiria’s network is based on L2 switching solutions and very few routers are used in the topology at all. Control and management is one of the highest concerns when dealing with the network. This is noticeable by looking at the fact that no routing protocol is used within it. All routing tables are updated manually with static routes.

Since the IPLeiria’s network topology consists of Cisco IOS equipment which, if it is running an older version of the OS, can easily be updated to support IPv6 the best solution for a transition mechanism is the implementation of a dual stack mechanism along with a well thought out addressing plan.

Chapter 5 Tunneling Tests

In this chapter we describe the test scenarios implemented to test some of the transition mechanisms that were previously referred. This chapter was developed early on in the work process and as such we were focused on tunneling and delivering IPv6 connectivity over an IPv4 only network.

Scenarios were implemented using Cisco IOS routers and computers running Windows 7, and Linux Ubuntu Operating Systems. Details of the setup are exposed in the different scenarios.

Routing was configured manually without the use of any automatic routing protocol, the reason behind this being that the IPLeiria's network is configure as such and we decided to approximate the lab tests as much as possible with the case study.

Note that this chapter is meant as a guideline to achieve IPv6 connectivity on a network that possesses some IPv6 incompatible equipment. This is not the case of the IPLeiria's network however we were not aware of this fact at the time and decided to include this chapter as a small guide to tunneling and IPv6 connectivity over IPv4. We have since abandoned this road and following the IPLeiria's IT center UARS request focused more on other IPv6 issues. Some of the work was left unfinished due to the lack of time, interest and necessity of the specific nature of the IT center's UARS request.

5.1 Test Methodology

Regarding IPv6 tunneling over IPv4 the choices between different types of tunnels are considerable. Some tunnel types have been deprecated while others are still in use. Some technologies serve a better purpose for a certain objective than others, for this chapter we picked three types of tunnels that serve different purposes and are used for different kinds of connectivity requirements.

6to4 tunnels are used to create a point-to-multipoint virtual link between IPv4 connected nodes, in this case Cisco routers. This technology can be used to create a tunnel with multiple end points which is useful to connect various IPv6 islands together as well as connect them with the IPv6 Internet through a gateway.

Manual Tunnels are used to create a virtual point-to-point link. These tunnels can be used to connect a single IPv6 node with another single IPv6 node and are mainly used to connect two routers with each other. By doing this it is possible to connect two IPv6 islands together. This kind of tunnel is mainly used to connect separate sites together, like, for example, a local network of a certain organization located in Lisbon can be connected to the local network of the same organization located in Porto and by doing so create a link between both of them.

ISATAP tunnels were developed in order to bring IPv6 connectivity to isolated IPv6 nodes located inside a local network, or site. This kind of tunnel is auto configured on the client side using the prefix disseminated by the server and the nodes IPv4 address. They allow isolated IPv6 nodes to connect to an ISATAP server (usually a router) and from there achieve IPv6 connectivity.

Along with these three types of tunnels we have also tested and documented the use of two different tunnel broker providers. These tunnel broker providers act as an end point to a tunnel that we can configure on a client. Both the ones tested allow for a client side installed on a computer while one of them has more features like allowing the configuration of the tunnel end point in a Cisco router, a very interesting solution indeed.

5.2 IPv4 Tunneling Test Bed

The 6to4 Automatic Tunnel Test, Manual Tunnel Test and ISATAP Tunnel Test were designed and setup using the same basic network configuration.

5.2.1 6to4 Automatic Tunnels

A 6to4 tunnel utilizes special IPv6 addresses located in the 2002::/16 addressing space. The first 16 bits of the address are the number 2002 in hexadecimal form, and the following 32 bits are the original source IPv4 address converted into hexadecimal form. For example if we convert the 172.16.12.1 IPv4 address into its hexadecimal equivalent we get AC10:0C01 because 172 translates to AC, 16 translates to 10, 12 translates to 0C and 1 translates to 01. Since a 6to4 tunnel is not

a point-to-point link it does not require a destination address. These tunnels were developed to allow communication between IPv6 islands on top of an IPv4 backbone by automatically discovering the IPv6 address of the other end of the tunnel. The virtual interface tunnel IPv6 address is configured using the interfaces IPv4 address.

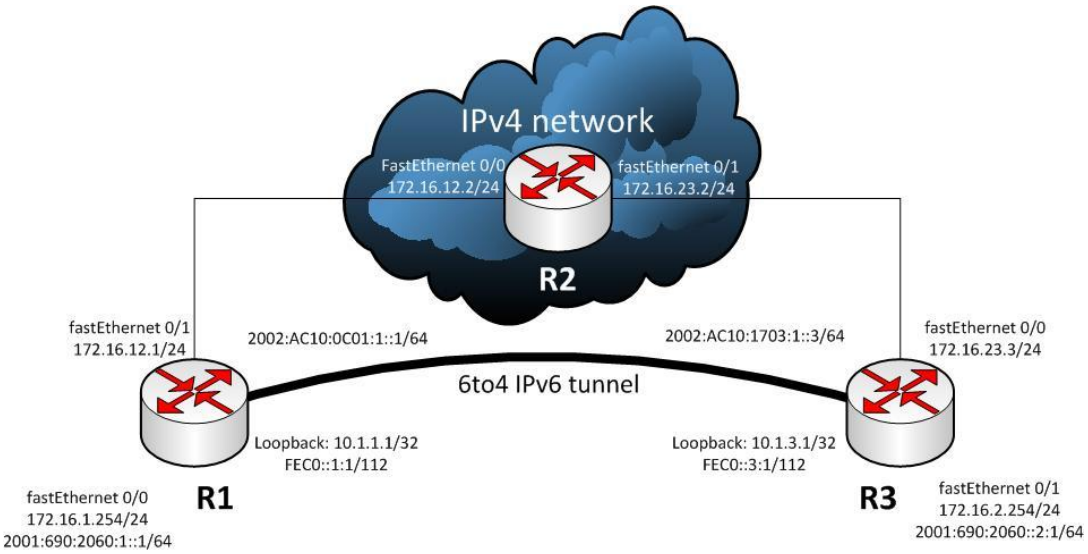


Figure 8 - IPv6 6to4 Tunnel Scenario Scheme

In this test scenario we configured a 6to4 tunnel according to Figure 8. Routers R1 and R3 are configured with a 6to4 tunnel to provide IPv6 connectivity between them and their loopback interfaces.

The first step is to establish basic IPv4 connectivity between all interfaces making sure R1 and R3 can both ping all the interfaces in the network. Also configure the IPv6 addresses for the physical interfaces and loopback interfaces.

Secondly the tunnel interface must be configured in both R1 and R3 as well setting up the IPv6 static routes in both routers so they can forward packets to each other.

It is important to note that in this type of tunnels the end point of the tunnel on the other side is not specified. This is determined automatically according to the IPv4 destination address (which is inserted in the tunnel other end-point's IPv6 address in hexadecimal notation) inserted into the routing table.

R1 config:

```
R1(config)# interface tunnel 0
R1(config-if)# tunnel mode ipv6ip 6to4
R1(config-if)# ipv6 address 2002:AC10:0C01:1::1/64
R1(config-if)# tunnel source fastethernet0/1
R1(config-if)# exit
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 route 2002::/16 tunnel0
R1(config)# ipv6 route FEC0::3:0/112 2002:AC10:1703:1::3
R1(config)# ipv6 route 2001:690:2060:2::/64
2002:AC10:1703:1::3
```

R3 config:

```
R3(config)# interface tunnel 0
R3(config-if)# tunnel mode ipv6ip 6to4
R3(config-if)# ipv6 address 2002:AC10:1703:1::3/64
R3(config-if)# tunnel source fastethernet 0/1
R3(config-if)# exit
R3(config)# ipv6 unicast-routing
R3(config)# ipv6 route 2002::/16 tunnel0
R3(config)# ipv6 route FEC0::1:0/112 2002:AC10:C01:1::1
R3(config)# ipv6 route 2001:690:2060:1::/64
2002:AC10:C01:1::1
```

When trying to ping R3 from R1 we get, with Wireshark, the packet trace in Figure 9. Notice how the IPv6 packet comes encapsulated inside the IPv4 packet whose Protocol Field reads 41 (IPv6). Wireshark is able to analyse the packet and read the IPv6 packet from within the IPv4 data field and display its header and payload. Notice how the packet travels through the IPv4 network even though an IPv6 address is being pinged. The Source address is R1's FastEthernet 0/1 address and the destination is R3's FastEthernet 0/0.

When R1 needs to forward the IPv6 packet it searches its routing table for a route to the desired destination, in this case pinging 2001:690:2060:2::1. R1's routing table has an entry to reach 2001:690:2060:2:: through 2002:AC10:1703:1::3 (R3's Tunnel 0 IPv6 address). Since 2002::/16 is the reserved prefix for 6to4 tunnels and AC10:1703 is equivalent to 172.16.23.3 R1 encapsulates the IPv6 packet in an IPv4 packet and routes it through the IPv4 network to the otherside of the tunnel where he knows the packet should be treated and re routed to its destination. In this case R3's own FastEthernet 0/1 which then replies through the same route in the oposite direction.

Figure 9 indicates the packets being traded in the Ping request / reply messages.

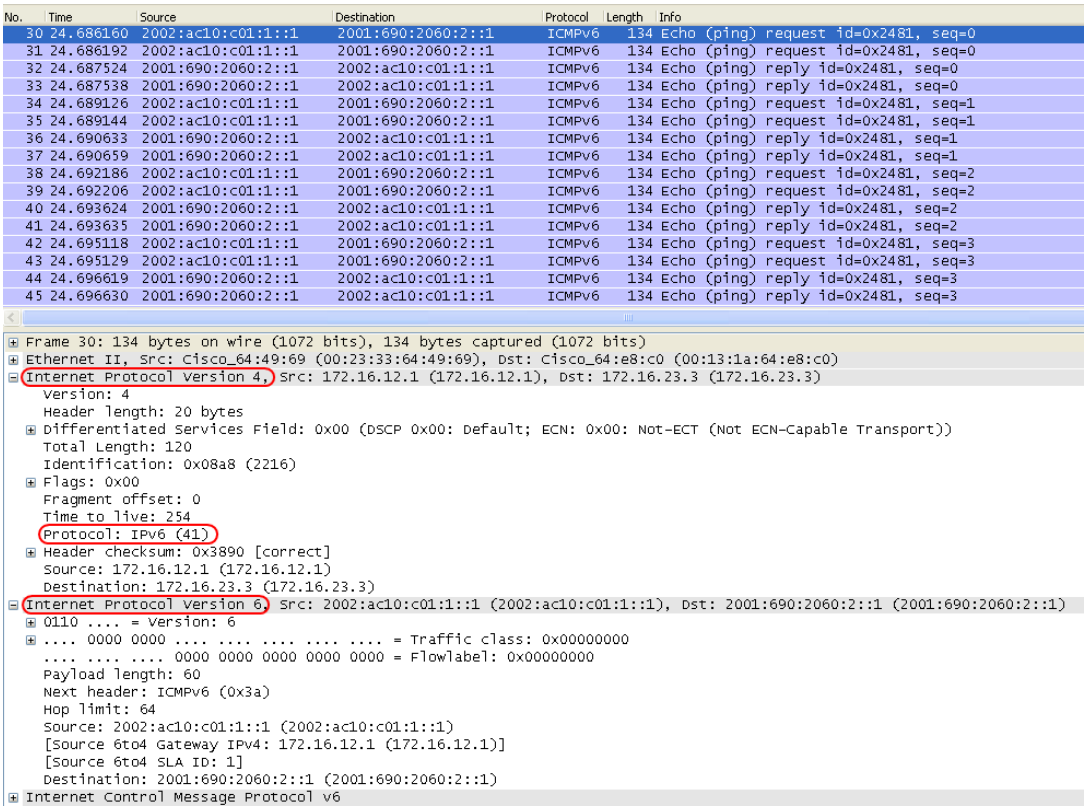


Figure 9 - 6to4 ping analysis

Using a tunnel configuration like this it is possible to configure a point-to-multi-point solution where two or more tunnel end points can be configured and packets are routed to the respective tunnel end point according to the routing table entries added to each router. It's a very interesting solution for a large network with lots of IPv4 only routers.

5.2.2 Manual Tunnels

An IPv6 manual tunnel is a type of tunnel that has hard-coded source and destination IPv4 addresses. Tunnel end point IPv6 addresses are not derived from the interfaces IPv4 address and are both members of the same subnet.

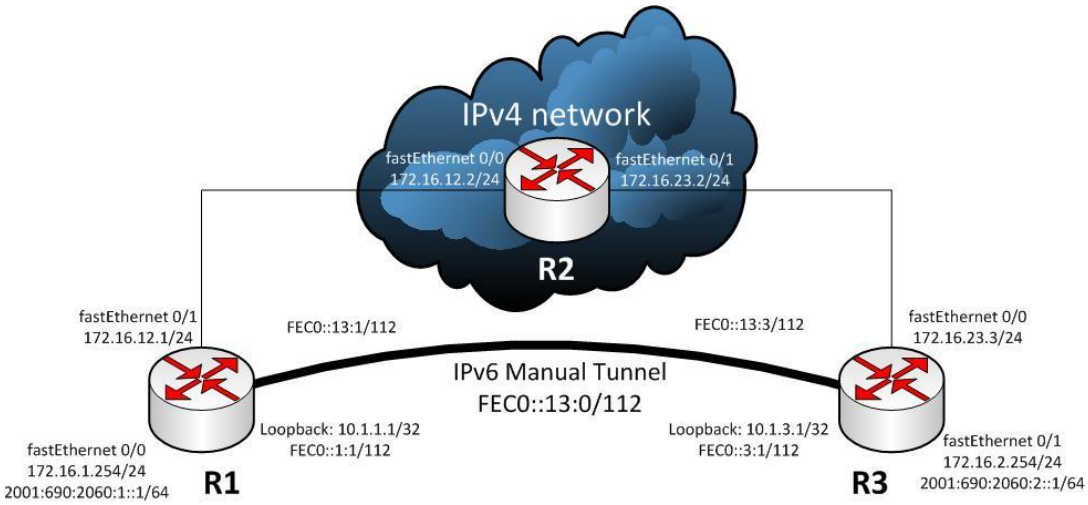


Figure 10 - IPv6 Manual Tunnel scenario

In this test scenario we configured an IPv6 Manual Tunnel according to Figure 10. Routers R1 and R3 are configured with an IPv6 Manual Tunnel to provide IPv6 connectivity between them and their loopback interfaces.

The first step is to establish basic IPv4 connectivity between all interfaces making sure R1 and R3 can both ping all the interfaces in the network and configure the IPv6 addresses for the physical interfaces and loopback interfaces.

Secondly the tunnel interface must be configured in both R1 and R3.

R1 config:

```
R1(config)# int tunnel0
R1(config-if)# tunnel mode ipv6ip
R1(config-if)# tunnel source fastethernet 0/1
R1(config-if)# tunnel destination 172.16.23.3
R1(config-if)# ipv6 add FEC0::13:1/112
R1(config-if)# exit
R1(config)# ipv6 unicast-routing
```

R3 config:

```
R3(config)# int tunnel0
R3(config-if)# tunnel mode ipv6ip
R3(config-if)# tunnel source fastethernet
0/0
R3(config-if)# tunnel destination
172.16.12.1
R3(config-if)# ipv6 add FEC0::13:3/112
R3(config-if)# exit
```

Thirdly instead of using static addresses it is possible to configure dynamic routing protocols such as OSPFv3 to advertise routes through the tunnel as if it was a regular physical interface.

Although the IPLeiria's network does not use dynamic routing we decide to configure OSPFv3 anyway just as a show case that it is possible to configure through a tunnel and that it works well in such configurations and a static route will work just as well.

R1 config:

```
R1(config)# interface loopback0
R1(config-if)# ipv6 ospf 1 area 0
R1(config-if)# interface tunnel0
R1(config-if)# ipv6 ospf 1 area 0
R1(config-if)# interface FastEthernet 0/0
R1(config-if)# ipv6 ospf 1 area 0
```

R3 config:

```
R3(config)# interface loopback0
R3(config-if)# ipv6 ospf 1 area 0
R3(config-if)# interface tunnel0
R3(config-if)# ipv6 ospf 1 area 0
R3(config-if)# interface FastEthernet0/1
R3(config-if)# ipv6 ospf 1 area 0
```

Since both routers have the other end-point address configured through the tunnel destination command connectivity between both end points is achieved directly and the packets are routed to those IPv4 addresses where they are analyzed and treated accordingly.

Alternatively static routes could be inserted in each router with the destination to ping the other router's subnet.

R1 config:

```
R1(config)# ipv6 route 2001:690:2060:2::/64
```

R3 config:

```
R3(config)# ipv6 route 2001:690:2060:1::/64
```

When trying to ping R3 from R1 we get the, with Wireshark, the packet trace in Figure 11. Notice how the IPv6 packet comes encapsulated inside the IPv4 packet whose Protocol Field reads 41 (IPv6) just like with a previously tested 6to4 tunnel. Wireshark is able to analyse the packet and read the IPv6 packet from within the IPv4 data field and display its header and payload. Just like with a 6to4 tunnel the packet travels through the IPv4 network even though an IPv6 address is being pinged. The Source address is R1's FastEthernet 0/1 address and the destination is R3's FastEthernet 0/0 address (both tunnel sources configured in the tunnel end points).

When R1 needs to forward the IPv6 packet it searches its routing table for a route to the desired destination, in this case pinging 2001:690:2060:2::1. R1's routing table has an entry to reach 2001:690:2060:2:: through Tunnel 0. This route is automatically inserted into the routing table by the OSPF protocol (or alternitively set with an IPv6 static route command). Since the tunnel is a virtual point-to-point link (and as expected both end points' addresses are within the same subnet) packets are routed directly from R1's tunnel 0 source address (FEC0::13:1/112) to R3's tunnel 0 source address (FEC0::13:3/112). When the encapsulated IPv6 packets reach R3 the router de-capsulates them and re-routes them to their destination. In this case R3's own FastEthernet 0/1 which then replies through the same route in the oposite direction.

Figure 11 indicates the packets being traded in the Ping request / reply messages.

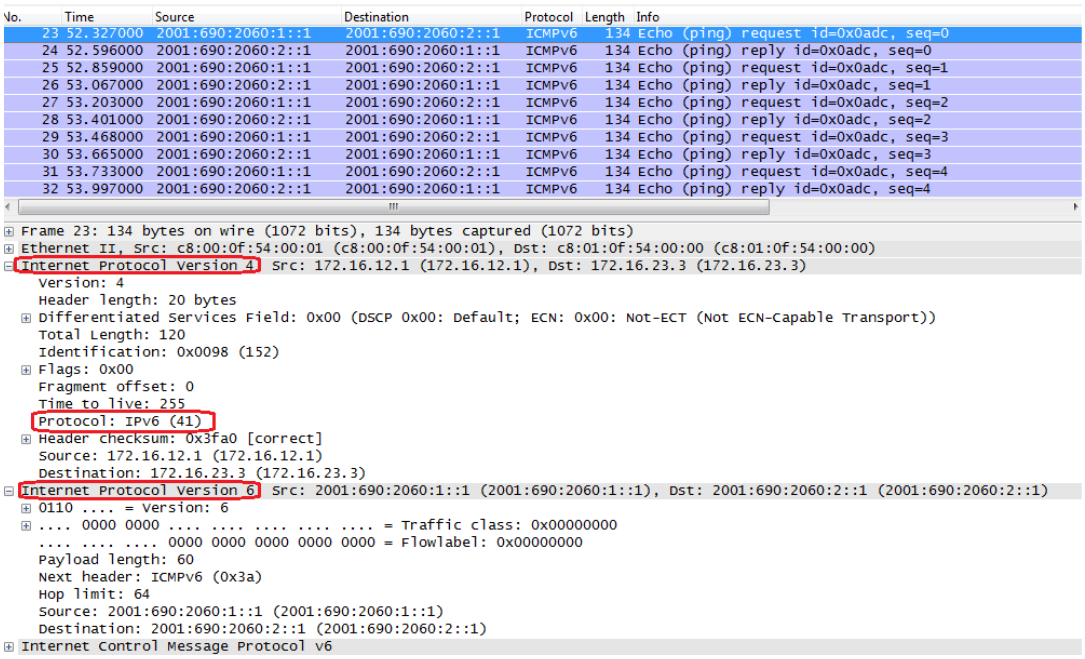


Figure 11 - Manual Tunnel ping analysis

Using a tunnel configuration like this it is possible to configure a virtual point-to-point link where two tunnel end points can be configured and packets routed to the opposing tunnel's end point according to the routing table entries added to both routers. This solution is used to create a connection between 2 routers, usually to connect an isolated IPv6 island to a larger IPv6 network like, for example, the native IPv6 Internet.

5.2.3 ISATAP

ISATAP (Infra-Site Automatic Tunnel Addressing Protocol) tunnels are used within a local site for IPv6 connectivity from a node to an edge router which provides IPv6 connectivity to the outside of the local network. This node can either be a single host computer or a router. For test purposes we have setup and implemented an ISATAP solution only in a routing equipment and not on a host computer due to the fact that the IPLeiria's network does not require such a solution. As soon as this fact was clear we switched our attention from tunneling solutions to other more interesting objectives. ISATAP provides an easy and efficient auto-configurable tunnel interface that communicates with a local ISATAP server (usually a router as is the case) that is charged with the task of providing IPv6 connectivity to the outside effectively creating an auto-configurable IPv6 connection to the global IPv6 network within the local network.

Figure 12 specifies the conditions of the test.

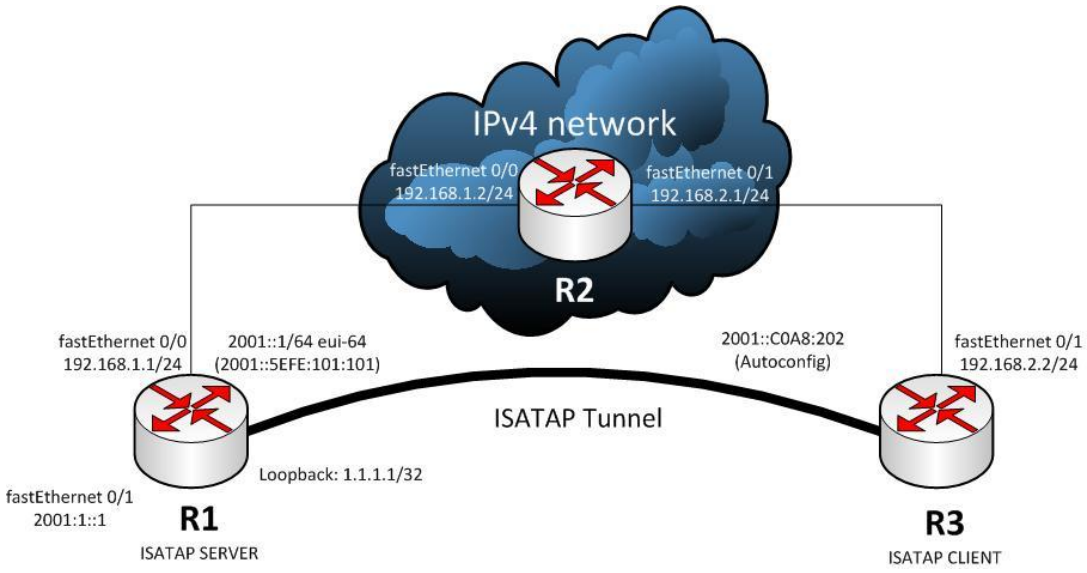


Figure 12 - ISATAP Tunnel scenario

The first step is to establish basic IPv4 connectivity between all interfaces making sure R3 can ping R1s loopback interface.

Secondly we setup both sides of the ISATAP tunnel.

R1 config:

```
R1(config)# ipv6 unicast-routing
R1(config)# interface tunnel0
R1(config-if)# ipv6 address 2001::1/64
eui-64
R1(config-if)# no ipv6 nd suppress-ra
R1(config-if)# tunnel source loopback 0
R1(config-if)# tunnel mode ipv6ip isatap
```

R3 config:

```
R3(config)# ipv6 unicast-routing
R3(config)# interface tunnel 0
R3(config-if)# ipv6 address autoconfig
R3(config-if)# ipv6 enable
R3(config-if)# tunnel mode ipv6ip
R3(config-if)# tunnel source fastEthernet
0/1
R3(config-if)# tunnel destination 1.1.1.1
```

The IPv6 address on the client side of the tunnel is now auto-configured and IPv6 connectivity between the ISATAP client and the ISATAP server has been achieved successfully.

Thirdly we add a route from R3 to R1's fastEthernet 0/1 subnet address in order to be able to reach that subnet from R3.

R3 config:

```
R3(config)# ipv6 route 2001:1::/64 2001::5EFE:101:101
```

When trying to ping R1's FastEthernet 0/1 address from R3, with Wireshark, we get the packet trace in Figure 13. Just like with a 6to4 and manual tunnels the packet travels through the IPv4 network even though an IPv6 address is being pinged. The Source address is R3's FastEthernet 0/1 IPv6 address which is autoconfigured using the announced prefix from R1's tunnel 0 and a node ID built using eui-64 (which basicaly means the hexadecimal notation of the interfaces IPv4 address) the destination is R1's FastEthernet 0/1 address.

When R3 needs to forward the IPv6 packet it searches its routing table for a route to the desired destination, in this case pinging 2001:1::1. R3's routing table has an entry to reach 2001:1:: through Tunnel 0. When the encapsulated IPv6 packets reach R3 the router de-capsulates them and re-routes them to their destination. In

this case R1's own FastEthernet 0/1 which then replies through the same route in the oposite direction.

Figure 13 indicates the packets being traded in the Ping request / reply messages.

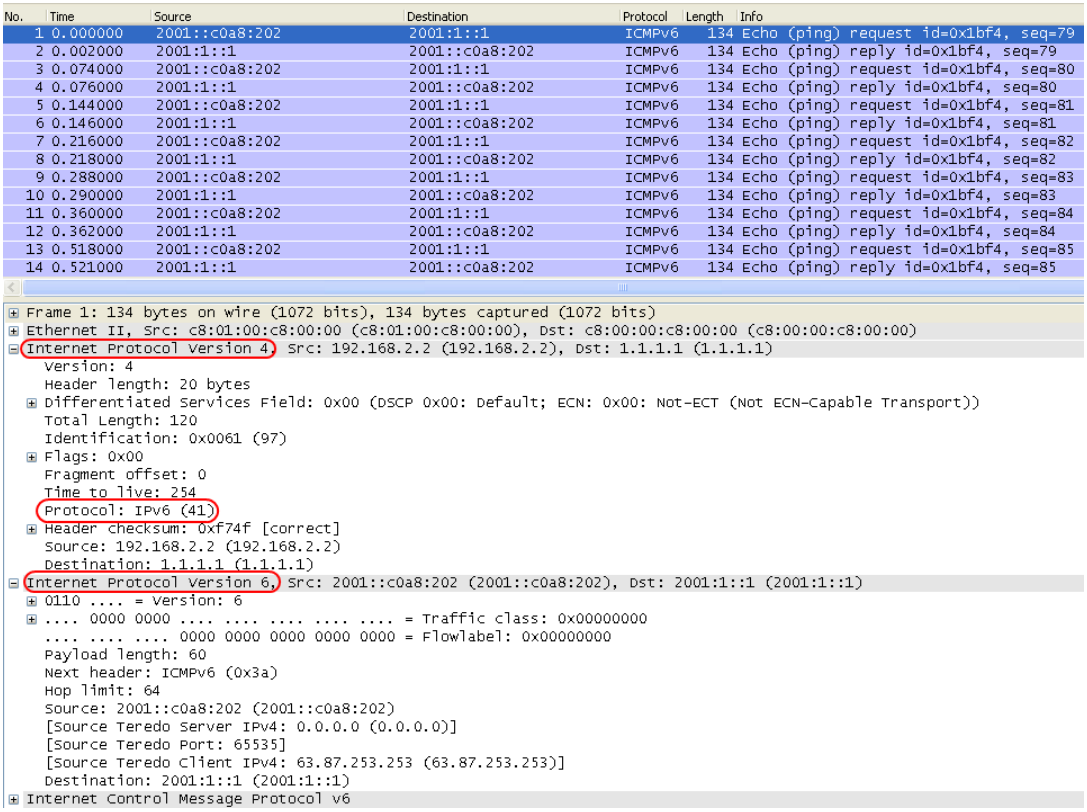


Figure 13 - ISATAP Wireshark ping analysis

Using a tunnel configuration like this it is possible to configure an automatic tunnel that, according to its standard, should only be used within a local site and never to connect a local network to the outside [36].

5.3 Tunnel Brokers

There are several Tunnel Broker solutions and we have configured and tested 3 of them. Two of them setup in Windows 7 and another setup in Ubuntu 11.04. We chose these Operating Systems due to their broad adoption and the fact that they are the most popular Operating Systems within the Campus amongst Computer Engineering students.

5.3.1 Hurricane Electric's free IPv6 tunnel broker service

This tunnel Broker Solution was configured and tested under windows 7.

In order to configure the tunnel using Hurricane Electric's Tunnel Broker (HETB) server we need to access the company's website and register an account. This solution allows for a multitude of clients to connect to the configured tunnel, from host Operating systems such as Windows 7 to Cisco IOS the Hurricane Electric Tunnel Broker is very flexible and supports a wide array of systems, however, since our aim with this kind of tunnels was to provide connectivity to a single isolated user on an IPv4 network and not to create a router to router link we worked with the operating system that is more prone to being used in the future at IPLeiria, Windows 7.

Also worthy of noting is the fact that due to the IPLeiria's network security policies this tunnel broker solution could not be configured within the IPLeiria's network, having been instead tested from a home connection. Due to this fact we were also unable to configure the tunnel in a Cisco IOS device for testing which would also have been an interesting solution.

Once logged in click on the "Create Regular Tunnel" link in the "User Functions" on the left side of the website and configure the "IPv4 Endpoint (Your side)" with the public address used by your network to communicate with the internet (usually the "You are viewing from:" IP address is the one you will want to configure in this field) and pick the Tunnel server you would like to connect to (you can use any or follow the recommended server).

Click the "Create Tunnel" button.

Figure 14 depicts the Configurations relative to the tunnel that was just created.

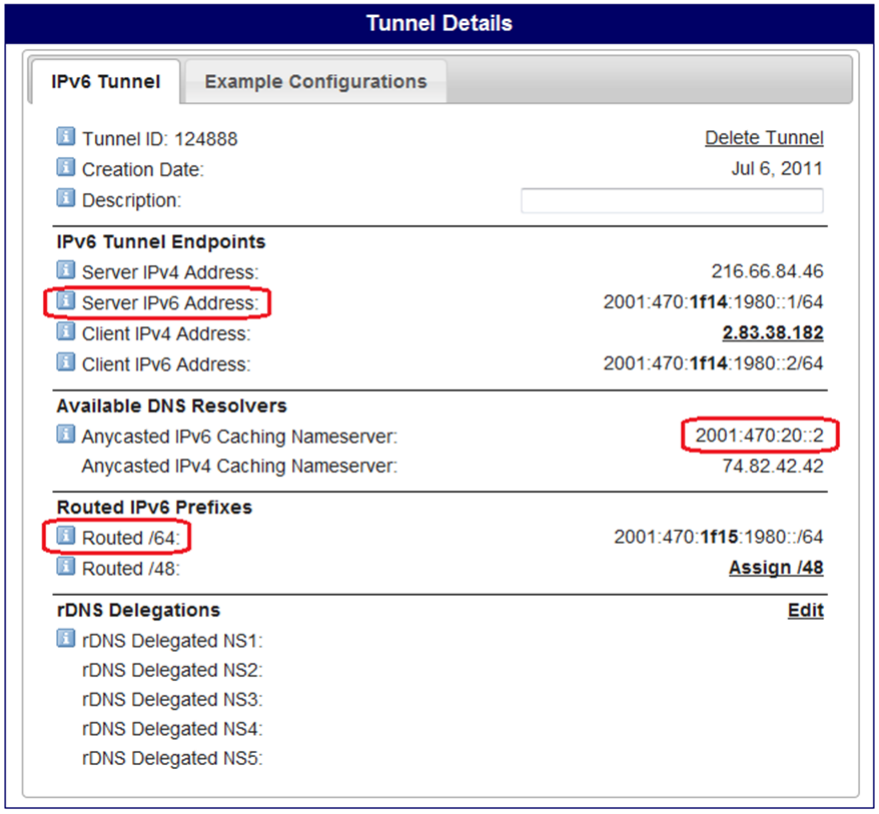


Figure 14 - Hurricane Electric Tunnel configurations

Before inserting the commands we have to manually address the IPv6 network configurations of our network card with an address from the "Routed /64" prefix so, for example, if the Routed /64 prefix is 2001:470:1f15:1980::/64 we can address our network card with, for example, the address 2001:470:1f15:1980::2/64. Set the gateway address to the "Server IPv6 Address" that you were given, for example, 2001:470:1f14:1980::1/64. Manually configure the DNS server to the provided address which should be 2001:470:20::2.

Figure 15 depicts the Example Configurations window which possesses the commands to be imputed on the client side to configure the tunnel end point.

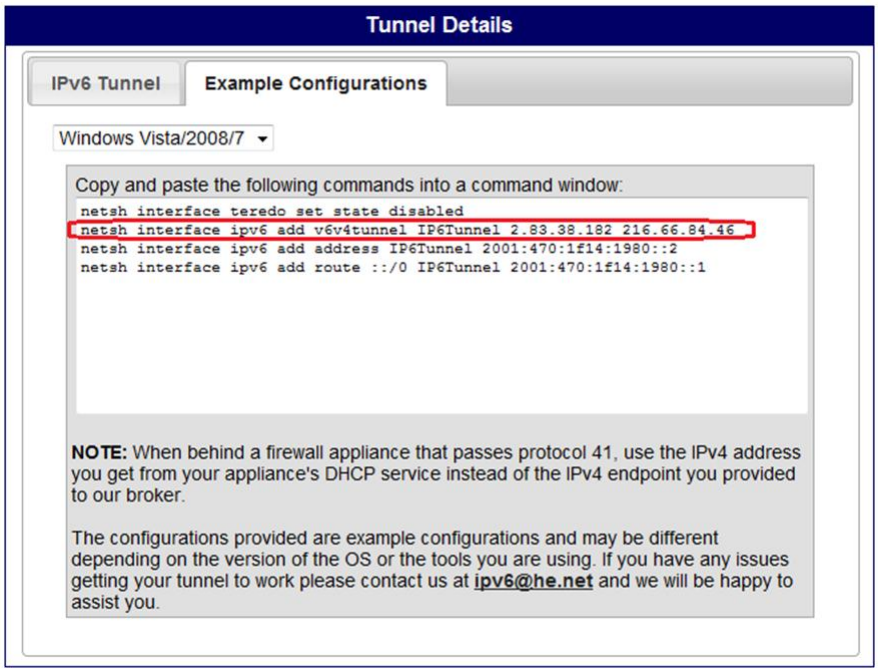


Figure 15 - Hurricane Electric Tunnel Example Configurations

Click on the Example Configurations tab and select the operating system you would like to configure the tunnel in, in this example Windows 7. Before inputting the commands be sure to swap the public address you inputted back when you created the tunnel with your local private address, for example, the netsh interface ipv6 add v6v4tunnel IP6Tunnel 2.83.38.182 216.66.84.46 command must be switched to the netsh interface ipv6 add v6v4tunnel IP6Tunnel <private address> 216.66.84.46 command.

Finally, after all these configurations come into effect ping http://ipv6.google.com/ and success. IPv6 connectivity has been established.

Figure 16 indicates the packets being traded in the Ping request / reply messages.

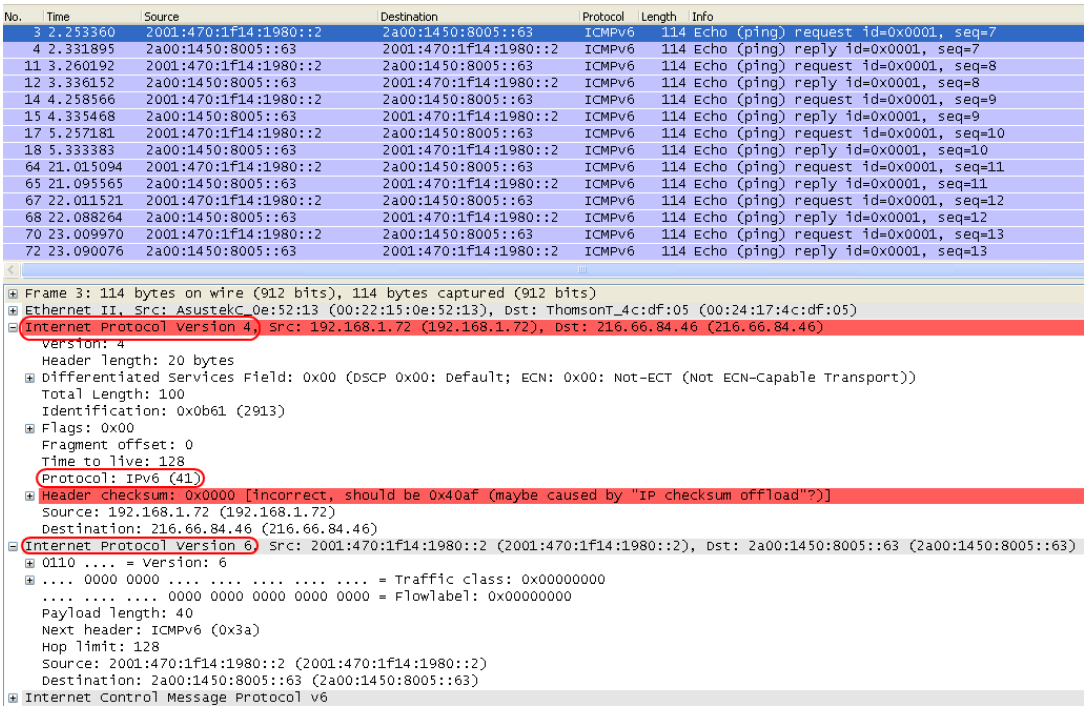


Figure 16 – HETB ping analysis

Regarding the checksum offload error Wireshark's homepage documentation explains it by stating that the network packet is transmitted to Wireshark before the checksum calculation is done which results in an empty checksum field being shown as invalid. However the packet is actually sent with a checksum field, it is merely calculated after it is "sniffed" by Wireshark [40].

Even though there seems to be a problem with the echo request IPv4 checksum the process follows along fine, IPv6 connectivity is achieved and ipv6.google.com is routable from the local network.

It is worth mentioning that because of the encapsulating and de-capsulation process as well as the fact that the IPv4 packets need to travel all the way to the tunnel broker gateway in order to be de-capsulated and only then do they follow their way to their destination the latency of this kind of connection will always be higher than a native IPv6 connection. However at the time of the test the results were rather surprising in a good way with an average latency of 77ms. Here follows the related ping and trace route commands (Figure 17).

```
C:\Users\Tiago>ping -6 -t ipv6.google.com

A fazer ping para ipv6.l.google.com [2a00:1450:8005::63] com 32 bytes de dados:
Resposta de 2a00:1450:8005::63: tempo=80ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=80ms
Resposta de 2a00:1450:8005::63: tempo=77ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=83ms
Resposta de 2a00:1450:8005::63: tempo=77ms
Resposta de 2a00:1450:8005::63: tempo=78ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=77ms
Resposta de 2a00:1450:8005::63: tempo=75ms
Resposta de 2a00:1450:8005::63: tempo=77ms
Resposta de 2a00:1450:8005::63: tempo=77ms
Resposta de 2a00:1450:8005::63: tempo=75ms
Resposta de 2a00:1450:8005::63: tempo=81ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=75ms
Resposta de 2a00:1450:8005::63: tempo=78ms
Resposta de 2a00:1450:8005::63: tempo=79ms
Resposta de 2a00:1450:8005::63: tempo=75ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=76ms
Resposta de 2a00:1450:8005::63: tempo=76ms

Estatísticas de ping para 2a00:1450:8005::63:
    Pacotes: Enviados = 26, Recebidos = 26,
             Perdidos = 0 (perda: 0%),
Tempo aproximado de ida e volta em milissegundos:
    Mínimo = 75ms, Máximo = 83ms, Média = 77ms
Control-C
^C
C:\Users\Tiago>tracert ipv6.google.com

A rastrear a rota para ipv6.l.google.com [2a00:1450:400b:c00::68]
até um máximo de 30 saltos:

  1    82 ms    81 ms    80 ms  2001:470:1f14:1980::1
  2    71 ms    71 ms    72 ms  gige-g2-20.core1.ams1.he.net [2001:470:0:7d::1]

  3    73 ms    71 ms    71 ms  pr61.ams04.net.google.com [2001:7f8:1::a501:5169
:1]
  4    72 ms    72 ms    77 ms  2001:4860::1:0:4b3
  5    94 ms    95 ms   100 ms  2001:4860::1:0:2746
  6    94 ms    98 ms    94 ms  2001:4860::2:0:277a
  7   108 ms   111 ms   101 ms  2001:4860:0:1::395
  8    94 ms    98 ms    94 ms  dy-in-x68.1e100.net [2a00:1450:400b:c00::68]

Rastreio concluído.
```

Figure 17 – HETB latency test

Obviously these results will depend on connection speed and server load, as well as other factors, but all in all it is a very interesting solution for IPv6 connectivity especially for single nodes and isolated users that require or demand IPv6 Internet connectivity.

It would have been interesting to setup and configure a tunnel broker configuration from a routing equipment to the IPv6 internet however due to the IPLeiria's network security policies it is not possible to configure such solution without access to the gateway equipment and security policies which we unfortunately did not possess at the time of developing this project. Still the possibility is there to route all the IPv6 traffic from a local network through this tunnel broker solution.

5.3.2 gogo6's freenet6 service for windows

Configured and tested under windows 7.

For windows there is a graphics based client application (Figure 18) that is easy to install and run that can be used by virtually anyone without any kind of specific computer knowledge.

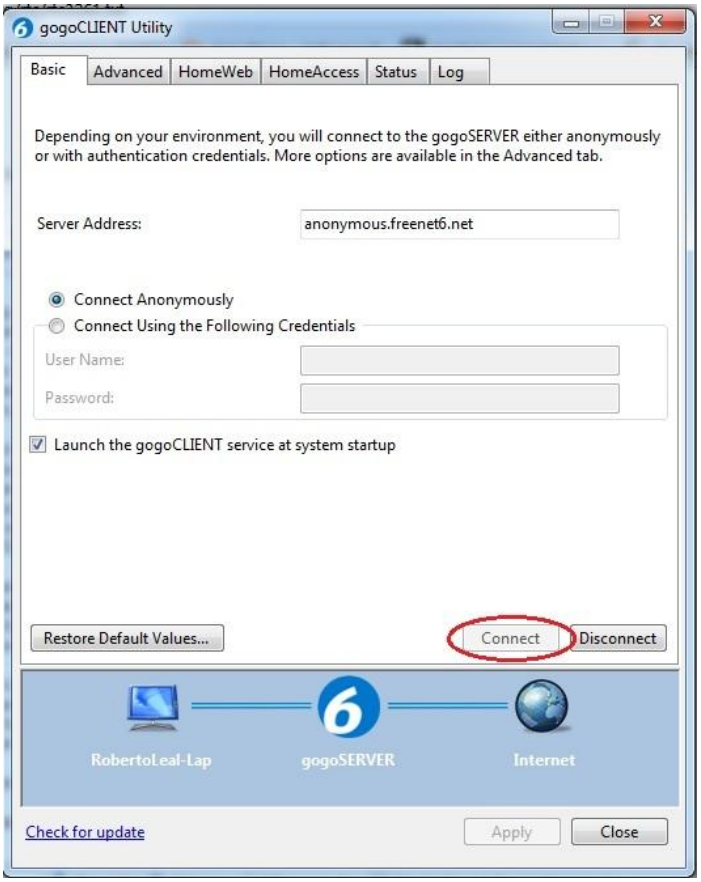


Figure 18 - gogoCLIENT Utility Window

To create, setup and run this tunnel broker solution access gogonet's website and sign up for a free account. After signing click the Freenet6 link or access the url directly http://gogonet.gogo6.com/page/freenet6-ipv6-services. click on Download and download the gogoCLIENT for Windows (as of this writing current version is 1.2). Install and run the gogoCLIENT Utility. Click connect and after a few seconds you will have IPv6 connectivity with the IPv6 internet tunneled through the gogo6SERVER.

Figure 19 shows the gogoClient Status Tab.

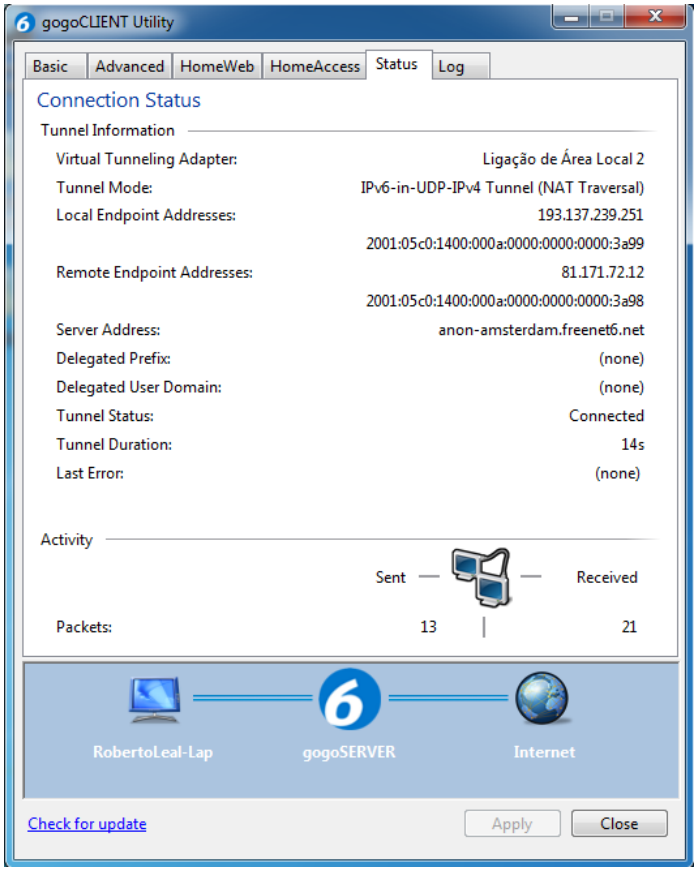


Figure 19 - gogoCLIENT Utility Status Window

Using the basic anonymous connection there is no need to register an account with the server and we are attributed the "Local Endpoint Address" and IPv6 connectivity with the "Remote Endpoint Address" is achieve along with IPv6 internet connectivity.

Gogo6 actually installs a brand new network interface in your network connections. It works by encapsulating IPv6 packet data into IPv4 UDP packets and sending them through the IPv4 network.

Figure 20 indicates the packets being traded in the Ping request / reply messages.

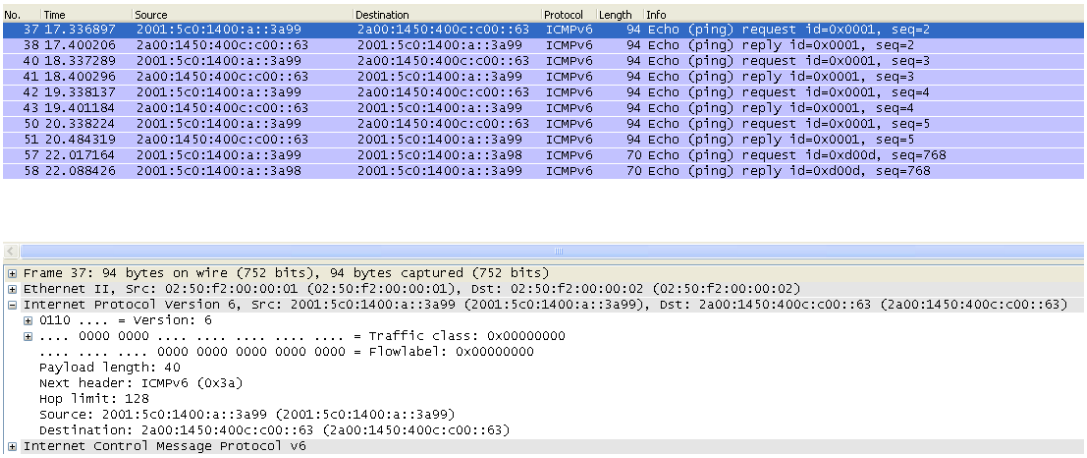


Figure 20 - gogoClient interface Wireshark ping (Windows7)

This is why a wireshark capture from gogo6's own network interface only shows IPv6 packet as though it were a native IPv6 connection when in fact if we repeat the ping process and instead capture the packets from the real physical interface we can see the IPv4 packets being routed to the IPv4 Remote Endpoint Address via UDP.

Figure 21 shows the IPv4 UDP packets in which the IPv6 packets are encapsulated.

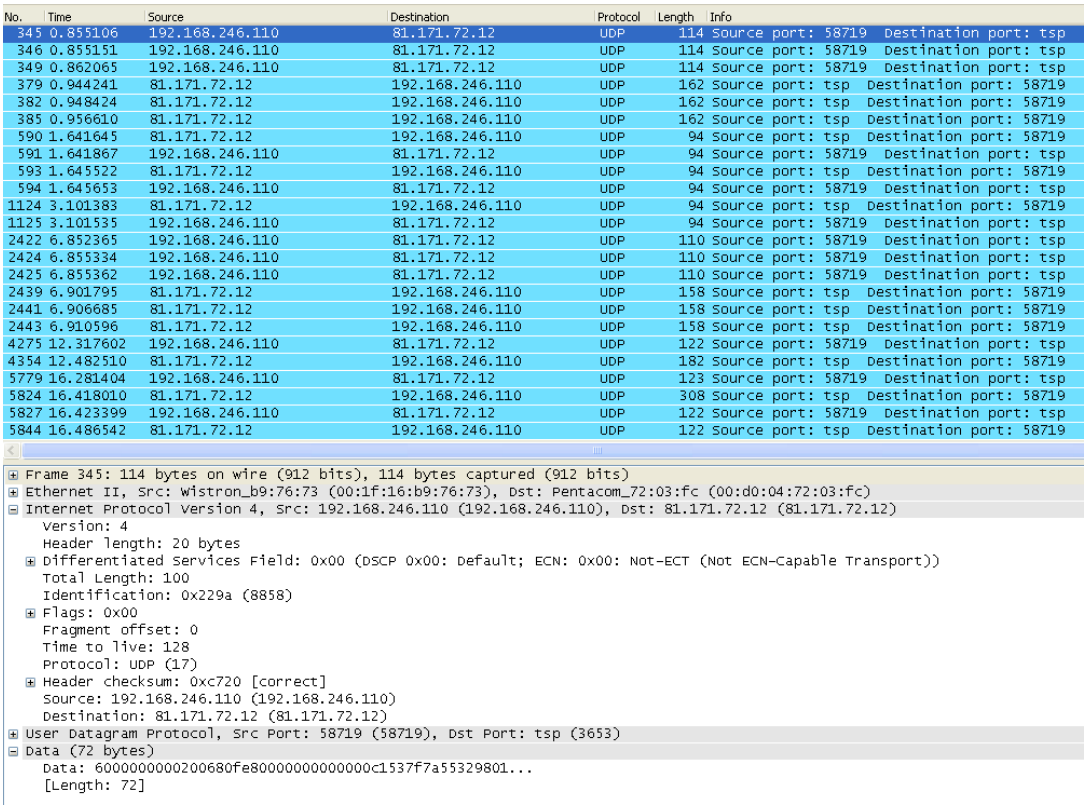


Figure 21 -Local network (gogo6) Interface wireshark ping (Windows7)

Looking at this wireshark packet trace it is possible to notice the behavior of the packets being traded with the local ipv4 address and the gogo6's remote ipv4 address (81.171.72.12) hence explaining why the virtual interface only seems to route ipv6 packets, all these packets are actually being encapsulated and routed through the IPv4 network over UDP on the physical network interface.

5.3.3 gogo6's freenet6 service for linux

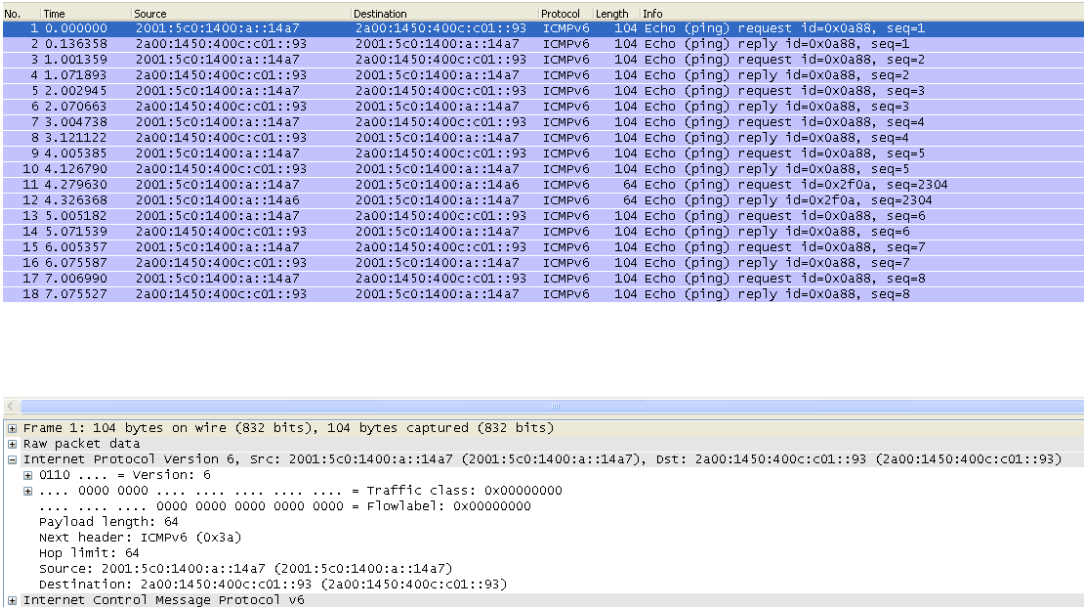
Configured and tested under Ubuntu 11.04

For Ubuntu it is just as easy, even though there is no GUI for it all you need to do is download and install the package with the command ***apt-get install gogoc*** and after installing run the command ***sudo /etc/init.d/gogoc start*** this will start the service and automatically configure your tunnel allowing you to get IPv6 connectivity with the IPv6 internet. You can now start using your IPv6 connectivity. Try pinging ipv6.google.com!

Notice that gogoC (gogo6's linux application) works in the same way as the windows gogoClient version, that is it creates the virtual interface and while this interface only seems to route IPv6 packets the physical network adapter interface is

actually encapsulating the IPv6 packet in the IPv4 packet data field and sending it inside a UDP packet on top of the IPv4 network to the gogo6 server IPv4 address.

Figure 22 indicates the packets being traded in the Ping request / reply messages.



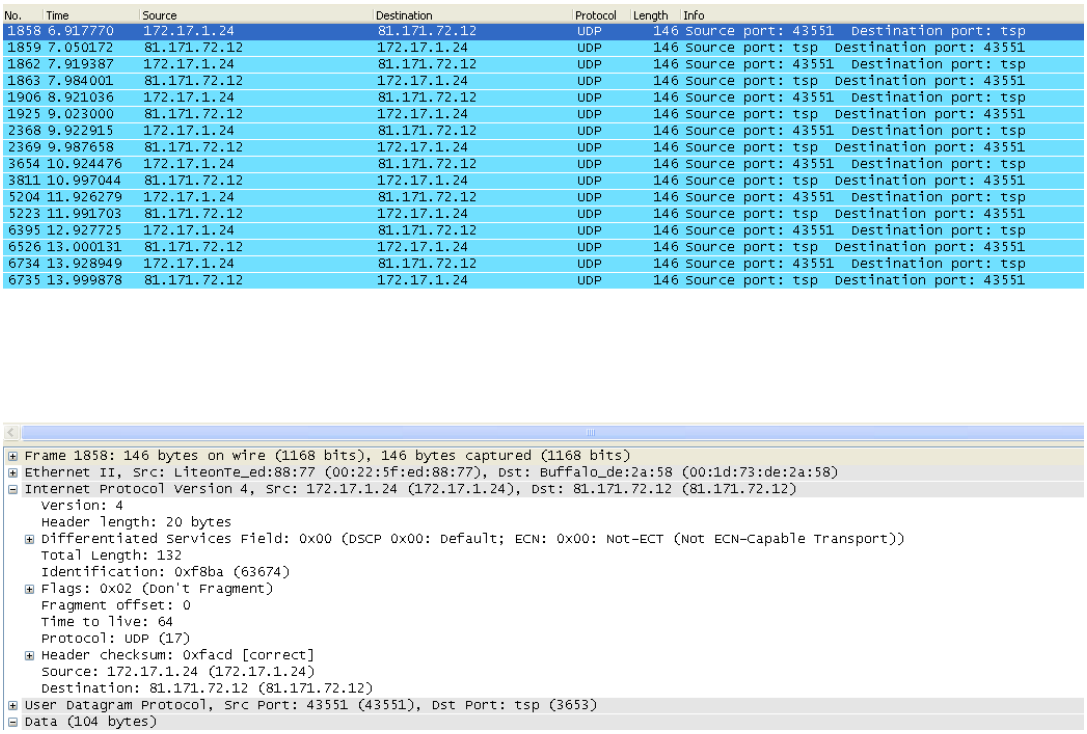
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=1
2	0.136358	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=1
3	1.001359	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=2
4	1.071893	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=2
5	2.002945	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=3
6	2.070663	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=3
7	3.004738	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=4
8	3.121122	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=4
9	4.005385	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=5
10	4.126790	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=5
11	4.279630	2001:5c0:1400:a::14a7	2001:5c0:1400:a::14a6	ICMPv6	64	Echo (ping) request id=0x2f0a, seq=2304
12	4.326368	2001:5c0:1400:a::14a6	2001:5c0:1400:a::14a7	ICMPv6	64	Echo (ping) reply id=0x2f0a, seq=2304
13	5.005182	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=6
14	5.071539	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=6
15	6.005357	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=7
16	6.075587	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=7
17	7.006990	2001:5c0:1400:a::14a7	2a00:1450:400c:c01::93	ICMPv6	104	Echo (ping) request id=0x0a88, seq=8
18	7.075527	2a00:1450:400c:c01::93	2001:5c0:1400:a::14a7	ICMPv6	104	Echo (ping) reply id=0x0a88, seq=8

```
⊞ Frame 1: 104 bytes on wire (832 bits), 104 bytes captured (832 bits)
⊞ Raw packet data
⊟ Internet Protocol Version 6, Src: 2001:5c0:1400:a::14a7 (2001:5c0:1400:a::14a7), Dst: 2a00:1450:400c:c01::93 (2a00:1450:400c:c01::93)
  ⊞ 0110 .... = Version: 6
  ⊞ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 64
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source: 2001:5c0:1400:a::14a7 (2001:5c0:1400:a::14a7)
    Destination: 2a00:1450:400c:c01::93 (2a00:1450:400c:c01::93)
⊞ Internet Control Message Protocol v6
```

Figure 22 - gogoc interface Wireshark ping (Ubuntu 11.04)

Packets captured from the gogoc tunnel interface show themselves as being native IPv6 packets. This is due to the fact that when the OS places the packets in this interface they have already been de-capsulated from the IPv4 UDP packet where they were transferred from.

Figure 23 shows the IPv4 UDP packets in which the IPv6 packets are encapsulated.



No.	Time	Source	Destination	Protocol	Length	Info
1858	6.917770	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
1859	7.050172	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551
1862	7.919387	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
1863	7.984001	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551
1906	8.921036	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
1925	9.023000	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551
2368	9.922915	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
2369	9.987658	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551
3654	10.924476	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
3811	10.997044	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551
5204	11.926279	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
5223	11.991703	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551
6395	12.927725	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
6526	13.000131	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551
6734	13.928949	172.17.1.24	81.171.72.12	UDP	146	Source port: 43551 Destination port: tsp
6735	13.999878	81.171.72.12	172.17.1.24	UDP	146	Source port: tsp Destination port: 43551


```
⊞ Frame 1858: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits)
⊞ Ethernet II, Src: LiteonTe_ed:88:77 (00:22:5f:ed:88:77), Dst: Buffalo_de:2a:58 (00:1d:73:de:2a:58)
⊟ Internet Protocol Version 4, Src: 172.17.1.24 (172.17.1.24), Dst: 81.171.72.12 (81.171.72.12)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 132
    Identification: 0xf8ba (63674)
  ⊞ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
  ⊞ Header checksum: 0xfacd [correct]
    Source: 172.17.1.24 (172.17.1.24)
    Destination: 81.171.72.12 (81.171.72.12)
⊞ User Datagram Protocol, Src Port: 43551 (43551), Dst Port: tsp (3653)
⊟ Data (104 bytes)
```

Figure 23 - Local network (gogo6) Interface wireshark ping (Ubuntu 11.04)

As with Windows gogoc works by encapsulating IPv6 packets inside IPv4 packets and sending these through the IPv4 network inside a UDP packet. On either side of the tunnel they are de-capsulated and placed in the appropriate interface, be that the gogo6 tunnel broker (server) or the PC where gogoc is installed and running (client).

5.4 Conclusions

The transition mechanics addressed in this chapter were chosen because they allow an easy and effective temporary transition from an IPv4 network to an IPv6 one. Although the need to implement such mechanisms does not exist at the moment in the IPLeiria's network we can never be sure that it will always be so. In case something fails, for example the native IPv6 link to the FCCN router and consequently all the IPv6 internet connection goes down these mechanisms allow for a backup IPv6 over IPv4 configuration to exist.

We recommend that steps be taken in relation to tunnel broker configurations. Be mindful that these mechanics raise certain security issues by "hiding" the IPv6 packet inside the IPv4 packet being transmitted and therefore certain firewall policies that exist for IPv4 will obviously not be taken into effect since the IPv4 header might look "clean" upon inspection. This raises issues related with the IPv6 compatibility of certain firewalls and their ability to analyze protocol 41 packets. As well, even IPv6 policies might not be taken into effect being that the packet may still be encapsulated upon inspection. Be mindful of these issues and take the required steps to combat them. For example a good idea may be to block all the known public tunnel broker addresses if the network administrator is fearful they might be used with bad intent in mind [39].

6to4 tunnels should be used in a point-to-multi-point configuration where more than two IPv6 islands need connection between themselves. Manual Tunnels should be used when a point-to-point link is required to connect one single IPv6 island with a broader IPv6 network. ISATAP tunnels should be used within a local site to allow single nodes to achieve IPv6 connectivity with an ISATAP server that is charged with the task of achieving IPv6 connectivity. Tunnel brokers are mainly used to provide IPv6 Internet connectivity to a single node through the IPv4 internet although the option to configure them in network equipment is also available. Bringing IPv6 connectivity to a whole IPv6 island is also possible this way.

As seen above there are various different options to achieve the same end result which is global IPv6 connectivity. Table 4 shows the different types of tested tunnels along with a comparison of their most noticeable features. When picking which solution to use it is important to analyses the situation and evaluate which technology suits better our own needs.

Table 4 - Tunnel Type Comparison

	6to4	**Manual**	**ISATAP**	**Tunnel Broker**
Type	Point-to-multipoint	Point-to-point	Point-to-point	Point-to-point
Properties		Not Autoconfigurable	Site only	Autoconfiguration depends on implementation
Use	To connect more than two IPv6 islands	To connect two IPv6 islands	To allow single nodes to achieve IPv6 connectivity	To provide IPv6 Internet connectivity to a single node through the IPv4 internet

Worthy of note is the fact that since the objectives of the project evolved while it was under study and development, namely while this chapter was being worked on, the focus shifted from tunneled solutions to other work like the creation of an addressing plan and the implementation of an IPv6 VoIP server. This was influenced by the fact that the IPLeiria's network is fully consisted of equipment that supports IPv6 making tunneling solutions unnecessary and the needs of the UARS towards other requirements like the ones aforementioned.

Chapter 6 IPv6 Transmission Test

With this IPv6 transmission test we hope to be able to shed some light over the native IPv6 versus Tunneled IPv6 predicament.

We would also like to note that we have configured an IPv6 enabled FTP server on the ESTGs IPv6 server (the webserver where the ipv6.estg.ipleiria.pt page is stored). This FTP server is accessible over IPv4 and IPv6. In order to connect to it over IPv4 you can use any FTP client, for IPv6 connections we tried the newest version of Filezilla Client and were able to connect and download the files therein with no issues.

Table 5 shows the IPv4 and IPv6 address of the machine where the FTP server was configured.

Table 5- FTP server access

IPv4	IPv6
Ipv6.estg.ipleiria.pt	2001:690:2060::2

With this FTP server configured we would like to test and compare IPv4 with IPv6 over an FTP transmission and since the IPv6 test bed we have setup would not translate to real world values we instead opted for doing a comparison test over the Internet.

6.1 Comparison tests

Along with IPv6 transition mechanisms we decided to implement a simple comparison between IPv4, IPv6 encapsulated in IPv4 and native IPv6.

We have contacted a colleague of ours, student Marco Paulo Cova who owns a dual stacked website physically located in Germany and he was nice enough to provide us with a download link so we could test all three IP setups. IPv4, IPv6 encapsulated in IPv4 and native IPv6 were tested by downloading said file and capturing the transmission with Wireshark. The file is acessible at www.void.pt/a.gz for IPv4 and ipv6.void.pt/a.gz for IPv6.

However we do realize that any results that may come from the developed tests cannot be taken too seriously for it is impossible to compare all three scenarios with each other due to the fact that IPv4 might be routed through a totally different path than IPv6 to reach the same physical location. Other factors like the amount of users in one connection link or another can also drastically influence the results as well as other possible temporary issues.

The transferred file has 25 MBytes.

6.2 IPv4 Transmission

We began by downloading the file using the wired native IPv4 connection available in ESTG's A.S.0.2 IT Project Room and got the following results. We used Wireshark to capture the packets and calculate the following graphic (Figure 24); it shows the TCP transmission in Bytes per Tick in relation with time spent. By looking at it we can easily identify the time at which the TCP transmission was running.

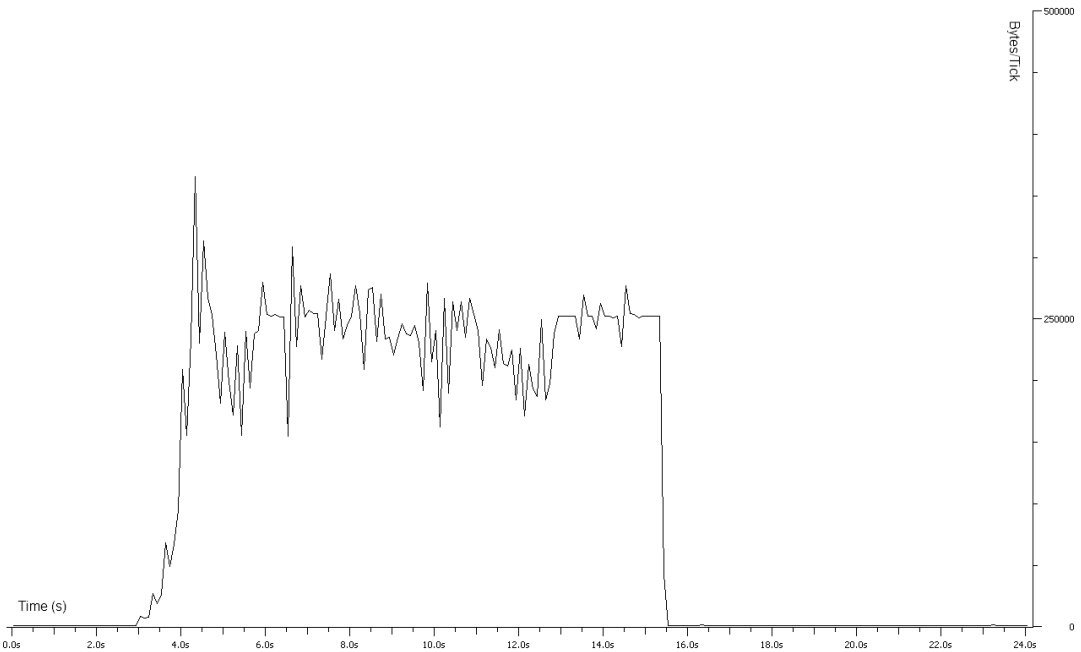


Figure 24 - IPv4 TCP transmission graphic

The following summary (Figure 25) details the average download rate of the TCP transmission during the download period. It shows an average transfer rate of around 12.5 MBits/sec and roughly a 17.5 seconds transmission time (Displayed column).

Traffic	Displayed
Packets	23757
Between first and last packet	17,563 sec
Avg. packets/sec	1352,691
Avg. packet size	1157,633 bytes
Bytes	27501897
Avg. bytes/sec	1565920,674
Avg. MBit/sec	12,527

Figure 25 - IPv4 TCP transmission summary

Just to make sure we followed the right stream and that the file size is correct we calculated the real size of the transfer (17.563*12.527)/8= 27.501 Mbytes which complies with the size of the file.

When calculating the overhead traffic generated from IPv4 we get 27.501 – 25 = 2.501Mbytes

Another thing to look at and analyze is the trace route to the server where the file is being transferred from (Figure 26). The number of jumps can also help as an indicator to the transfer, less jumps means less time wasted in rerouting and processing which translates in less latency time.

```
[root@ipv6 ~]# traceroute www.void.pt
traceroute to www.void.pt (188.40.118.195), 30 hops max, 40 byte packets
 1  192.168.0.254 (192.168.0.254)  0.245 ms  0.202 ms  0.205 ms
 2  193.137.239.254 (193.137.239.254)  0.632 ms  0.957 ms  0.516 ms
 3  ROUTER3.GE.Lisboa.fccn.pt (193.136.1.65)  4.693 ms  4.409 ms  4.405 ms
 4  ROUTER4.10GE.Lisboa.fccn.pt (193.137.0.4)  4.533 ms  4.518 ms  4.463 ms
 5  fccn.rt1.mad.es.geant2.net (62.40.124.97)  30.035 ms  30.030 ms  29.997 ms
 6  as2.rt1.gen.ch.geant2.net (62.40.112.25)  52.094 ms  52.090 ms  52.056 ms
 7  as0.rt1.mil.it.geant2.net (62.40.112.34)  59.496 ms  59.459 ms  59.421 ms
 8  mno-b1-link.telia.net (213.248.97.101)  59.563 ms  59.421 ms  59.391 ms
 9  ffm-bb1-link.telia.net (80.91.249.38)  68.249 ms  68.169 ms  68.150 ms
10  ffm-b2-link.telia.net (80.91.249.101)  68.566 ms ffm-b2-link.telia.net (80.
11  hetzner-ic-134650-ffm-b2.c.telia.net (213.248.92.82)  75.174 ms  75.215 ms
12  hos-bb1.juniper1.fs.hetzner.de (213.239.240.242)  79.912 ms  79.895 ms hos-
13  hos-tr1.ex3k10.rz10.hetzner.de (213.239.227.139)  80.520 ms hos-tr4.ex3k10.
14  hosting.void.pt (188.40.118.195)  73.327 ms  73.320 ms  73.326 ms
```

Figure 26 - IPv4 traceroute

So on the IPv4 connection available to us in the A.S.0.2 project room we get 14 hops from our client machine to the server where the file is located.

6.3 IPv6 Transmission

We began by downloading the file using the wired native IPv6 connection available in ESTG's A.S.0.1 aka L.C.A. and got the following results. We used Wireshark to capture the packets and calculate the following graphic (Figure 27) it shows the TCP transmission in Bytes per Tick in relation with time spent. By looking at it we can easily identify the time at which the TCP transmission was running.

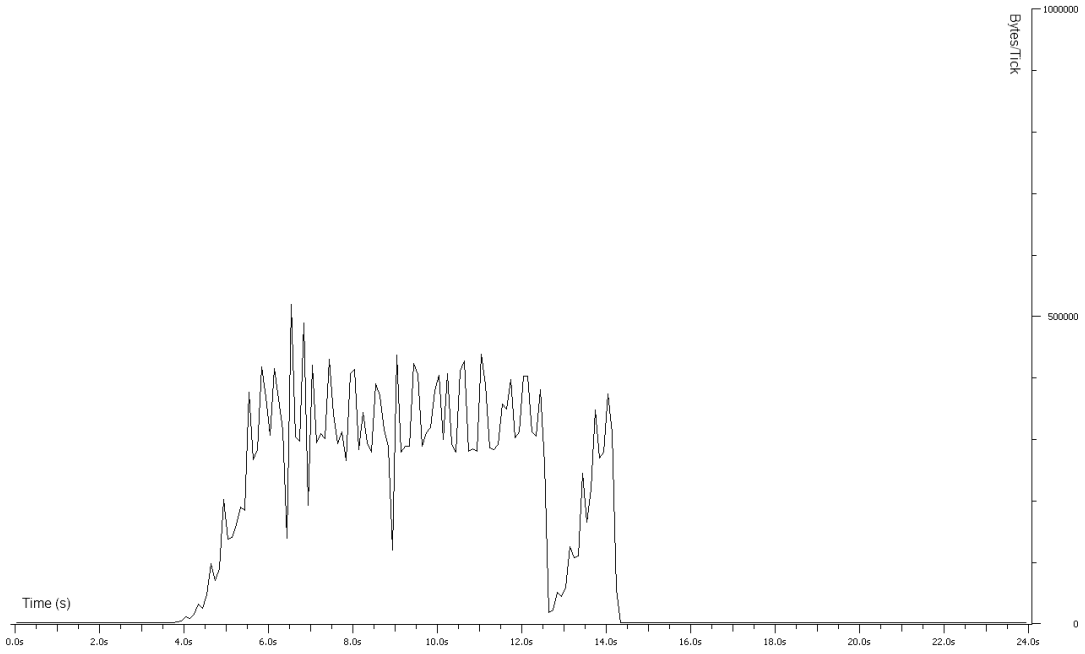


Figure 27 - IPv6 TCP transmission graphic

The following summary (Figure 28) details the average download rate of the TCP transmission during the download period. It shows an average transfer rate of around 8.3 MBits/sec and roughly a 26.6 seconds transmission time (Displayed column).

Traffic	Displayed
Packets	22616
Between first and last packet	26,641 sec
Avg. packets/sec	848,909
Avg. packet size	1233,401 bytes
Bytes	27894595
Avg. bytes/sec	1047045,160
Avg. MBit/sec	8,376

Figure 28 - IPv6 TCP transmission summary

Just to make sure we followed the right stream and that the file size is correct we calculate the real size of the transfer (26.641*8.376)/8= 27.893 Mbytes which complies with the size of the file.

When calculating the overhead traffic generated from IPv6 we get 27.893 – 25 = 2.893Mbytes

Another thing to look at and analyze is the trace route to the server where the file is being transferred from (Figure 29). The number of jumps can also help as an indicator to the transfer, less jumps means less time wasted in rerouting and processing which translates in less latency time.

```
C:\Windows\system32>tracert -6 www.void.pt

A rastrear a rota para void.pt [2a01:4f8:101:363::2]
até um máximo de 30 saltos:

  1     3 ms     6 ms    42 ms  2001:690:2060::1
  2     *        7 ms     3 ms  2001:690:2060::1
  3     8 ms     *        7 ms  ROUTER7.IPv6.fccn.pt [2001:690:810:14::1]
  4     7 ms     *        6 ms  ROUTER4.IPv6.10GE.Lisboa.fccn.pt [2001:690:800:2::4]
  5    31 ms    32 ms    39 ms  fccn.rt1.mad.es.geant2.net [2001:798:17:10aa::d]
  6    54 ms    54 ms    56 ms  as2.rt1.gen.ch.geant2.net [2001:798:cc:1201:1701::1]
  7    62 ms    62 ms    61 ms  as0.rt1.mil.it.geant2.net [2001:798:cc:1201:1e01::2]
  8    71 ms    73 ms    70 ms  mno-b1-link.telia.net [2001:2000:3081:15::1]
  9    72 ms    71 ms    72 ms  ffm-b2-v6.telia.net [2001:2000:3018:13::1]
 10    70 ms    70 ms    71 ms  hetzner-ic-134650-ffm-b2.c.telia.net [2001:2000:3080:18a::2]
 11    75 ms    76 ms    79 ms  hos-bb1.juniper2.rz10.hetzner.de [2a01:4f8:0:1::10:2]
 12    75 ms    75 ms    75 ms  hos-tr3.ex3k10.rz10.hetzner.de [2a01:4f8:0:10:3:a:10:10]
 13    74 ms    75 ms    75 ms  void.pt [2a01:4f8:101:363::2]

Rastreio concluído.
```

Figure 29 - IPv6 traceroute

So on the native IPv6 connection available to us in the L.C.A. test bed we get 13 hops from our client machine to the server where the file is located. Note that for some reason the IPLeiria's gateway is hoping to itself on the first hop; however this is not something we can control.

6.4 Tunneled IPv6 Transmission

We began by downloading the file using the wired IPv4 connection available in ESTG's A.S.0.2 IT Project Room to encapsulate our IPv6 connection. In this transmission we used the gogo6 tunnel broker we presented in the transition mechanisms chapter and got the following results. We used Wireshark to capture the packets and calculate the following graphic (Figure 30); it shows the TCP transmission in Bytes per Tick in relation with time spent. By looking at it we can easily identify the time at which the TCP transmission was running.

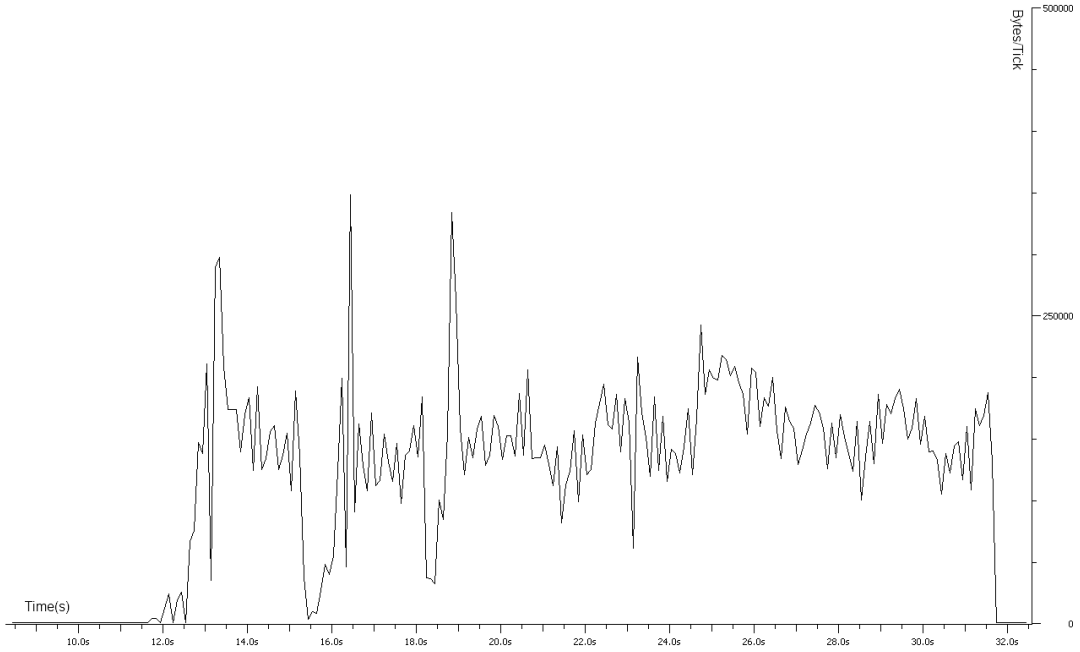


Figure 30 - Tunnelled IPv6 TCP transmission graphic

The following summary (Figure 31) details the average download rate of the TCP transmission during the download period. It shows an average transfer rate of around 8.8 MBits/sec and roughly a 25.6 seconds transmission time (Displayed column).

Traffic	Displayed
Packets	27832
Between first and last packet	25,186 sec
Avg. packets/sec	1105,063
Avg. packet size	1016,423 bytes
Bytes	28289088
Avg. bytes/sec	1123211,418
Avg. MBit/sec	8,986

Figure 31 - Tunnelled IPv6 TCP transmission summary

Just to make sure we followed the right stream and that the file size is correct we calculate the real size of the transfer (25.186*8.986)/8= 28.290 Mbytes which complies with the size of the file.

When calculating the overhead traffic generated from IPv4 we get 28.290 – 25 = 3.290Mbytes

Another thing to look at and analyze is the trace route to the server where the file is being transferred from. The number of hops can also help as an indicator to the transfer, less jumps means less time wasted in rerouting and processing which translates in less latency time. However since this IPv6 connection goes through a tunnel we have to add the IPv4 hops from our client to the tunnel end point and the hops from the tunnel end point to the server since the first hop from our client to the tunnel broker is equivalent to the whole sum of IPv4 hops from our client to the tunnel end point.

Figure 32 shows the IPv4 and IPv6 hop schematic for a tunneled IPv6 connection.

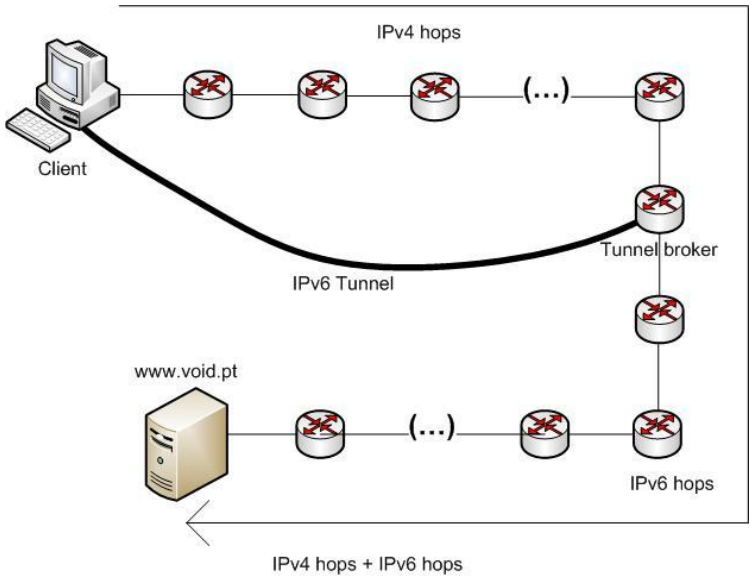


Figure 32 - tunnelled IPv6 traceroute scheme

The first step is to traceroute the IPv4 end of the tunnel broker we are using (Figure 33). According to the gogo6 client status tab the IPv4 address of the tunnel end is 81.171.72.12.

```
[root@ipv6 ~]# traceroute 81.171.72.12
traceroute to 81.171.72.12 (81.171.72.12), 30 hops max, 40 byte packets
 1  192.168.0.254 (192.168.0.254)  0.370 ms  0.249 ms  0.246 ms
 2  193.137.239.254 (193.137.239.254)  0.531 ms  0.881 ms  0.553 ms
 3  ROUTER3.GE.Lisboa.fccn.pt (193.136.1.65)  4.617 ms  4.461 ms  4.418 ms
 4  ROUTER10.10GE.Lisboa.fccn.pt (193.137.0.8)  4.682 ms  4.678 ms  4.644 ms
 5  Cogent.AS174.gigapix.pt (193.136.250.80)  5.080 ms  5.070 ms  5.035 ms
 6  te4-7.ccr01.opo01.atlas.cogentco.com (154.54.57.142)  9.747 ms te1-4.ccr01.opo0
 7  te1-4.ccr01.vgo01.atlas.cogentco.com (154.54.38.41)  12.230 ms te7-1.ccr01.vgo0
 8  te4-8.ccr01.bio01.atlas.cogentco.com (154.54.37.49)  22.691 ms te7-8.ccr01.bio0
 9  te4-2.ccr01.bod01.atlas.cogentco.com (154.54.38.69)  27.397 ms te3-2.ccr01.bod0
10  te0-7-0-1.ccr22.par01.atlas.cogentco.com (154.54.57.201)  35.888 ms te0-7-0-1.c
11  te0-1-0-5.mpd21.ams03.atlas.cogentco.com (130.117.51.61)  45.226 ms te0-1-0-5.c
12  te1-2.mpd03.ams03.atlas.cogentco.com (130.117.48.242)  45.137 ms  44.870 ms te1-
13  oxilion.demarc.cogentco.com (149.6.128.2)  45.478 ms  45.319 ms  45.377 ms
14  2-1.r2.am.hwng.net (69.16.191.101)  45.402 ms  45.363 ms  45.333 ms
15  2-3.colo-rx4.eweka.nl (81.171.115.242)  45.367 ms  45.355 ms  45.410 ms
16  ew-busdsl-81-171-72-12.eweka.nl (81.171.72.12)  45.761 ms  45.762 ms  45.693 ms
```

Figure 33 - Tunnelled IPv6 traceroute 1 (IPv4)

So on the IPv4 connection available to us from the L.C.A. room we get 16 hops from our client machine to the tunnel broker end point. Now we determine the IPv6 traceroute from the client to the server where the file is located (Figure 34).

```
C:\Windows\system32>tracert -6 ipv6.void.pt

A rastrear a rota para ipv6.void.pt [2a01:4f8:101:363::2]
até um máximo de 30 saltos:

  1    54 ms    97 ms    49 ms  2001:5c0:1400:a::1da
  2    48 ms    45 ms    47 ms  ve8.ipv6.colo-rx4.eweka.nl [2001:4de0:1000:a22::1]
  3    45 ms    45 ms    53 ms  9-1.ipv6.r2.am.hwng.net [2001:4de0:a::1]
  4    55 ms    55 ms    56 ms  amsix-gw.hetzner.de [2001:7f8:1::a502:4940:1]
  5    61 ms    59 ms    66 ms  hos-bb1.juniper2.rz10.hetzner.de [2a01:4f8:0:1::10:2]
  6    65 ms    61 ms    59 ms  hos-tr4.ex3k10.rz10.hetzner.de [2a01:4f8:0:10:4:a:10:10]
  7   116 ms    57 ms    58 ms  void.pt [2a01:4f8:101:363::2]
Rastreio concluído.
```

Figure 34 - Tunnelled IPv6 traceroute 2 (IPv6)

The first hop in IPv6 corresponds to the 16 hops in IPv4; our client is only able to perceive one IPv6 hop while, in reality, all of the packets are being routed over IPv4 to the tunnel broker.

In reality the tunneled IPv6 connection goes through 22 hops to reach its destination. However even with a higher hop count its transfer rate is still decent, even considering the longer path and the encapsulation / de-capsulation requirements it still manages to produce a decent result.

6.5 Conclusion

Relating to transfer speed, like we mentioned before, various different conditions may affect IPv4 / IPv6 comparisons. However one thing we can take from these tests is that tunneled IPv6 generates a higher overhead than IPv4. This was to be expected. Native IPv6 also seems to generate a slightly higher overhead than native IPv4, however this can also be affected by the same conditions that confine the speed comparison for example if the path MTU is low for IPv6 links then it will generate higher overhead than IPv4.

Theory states that if both Internet Protocol versions share the same physical link properties to the destination address (such as hop number

From this test we can conclude that even following a larger path as well as having to deal with the whole encapsulation / de-capsulation predicament a tunneled IPv6 solution is still robust. If the need to access the IPv6 internet arises a tunneled solution should make a decent backup connection if for some reason the user is unable to connect to the native IPv6 Internet.

Table 6 shows a value comprarison for the transmission test results we achieved

Table 6 - IPv4, IPv6 and Tunneled IPv6 comparison values

	IPv4	**IPv6**	**Tunneled IPv6**
Latency	73ms	75ms	77ms
Transmission Rate	12.527MBit/s	8.376MBit/s	8.986MBit/s
Overhead	10.004%	11.572%	13.16%

Tunneled IPv6's setup is very hard to analyze because on top of not being able to control the conditions of the IPv4 links between the host and the tunnel broker we also don't have access to the configurations on the tunnel broker side, for example, maximum bandwidth might be limited to a certain amount per stream.

Chapter 7 Addressing Plan

An Addressing Plan is a list of addressing rules and concepts adopted when projecting a relatively large network. In this chapter we will focus on developing a simple and effective IPv6 addressing plan to be applied to the IPLeiria's network in a dual stack configuration alongside the existing IPv4 addressing plan effectively implementing a stable and transparent transition mechanism. The rules and concepts adopted will be explained and a reason as to why that adoption was taken shall, as well, be presented. An alternative addressing plan and options are also provided as well as the means to build and improve upon the original addressing plan. Scalability of such an addressing plan is simple and adding more options in the future is relatively fast and easy to achieve. [41]

7.1 Why an addressing plan?

Private IPv4 network addressing plans have, for a long time, been using the private IPv4 address scope which was one of the "patch" solutions developed for IPv4 to enhance this protocol's lifetime. Along with NAT this technique became routine and nowadays the majority, if not all, networks adopt this methodology.

With the adoption of IPv6 the idea of site-local addresses was introduced. These addresses would be the IPv6 equivalent to the IPv4 private address range. However these addresses have been abandoned and their use deprecated. The humongous amount of addressing space available in IPv6 makes the need for such addresses obsolete thus allowing all hosts to be routable from any point in the network: all nodes possess a public address, accessible and routable from anywhere in the internet.

Due to the various facts that have already been stated in this report regarding IPv4 address exhaustion IPv4 addressing plans are based on efficient address assignment. However, if an organization or institution of decent size applies for an IPv6 addressing range at many Local Internet Registries (LIRs) around the world they

are usually assigned a /48 IPv6 prefix which translates to 2^80 available addresses. Since this number is so vastly superior to the IPv4 alternative where sometimes only a small handful of public addresses are assigned to the same organization or institution the need for efficiency tends to decrease significantly and this is the strongest reason why it is advised to adopt a simple and easy to read IPv6 addressing plan. A system in which addresses are assigned based on location and/or use type.

7.2 Benefits to an addressing plan

By grouping IPv6 address ranges following a set of effective and logic rules we are able to take several benefits out of an IPv6 addressing plan.

The first of which is visibility. By sorting addresses based on location or use type it becomes easier to detect where that address was used and what type of user it belonged to at the time. This will be explained in the following pages. Other advantages include the additional ease with which security policies such as access lists and firewall policies can be implemented by, for example, aggregating various routes into one /64 prefix. This also enables more efficient network management and can result in more compact and better performing equipment's routing table by routing /64 prefix networks instead of more specific ones. Add to these features the fact that an efficient addressing plan is scalable, that it can expand to, for example, include new use types and locations and it becomes extremely easy to perceive how fruitful an IPv6 addressing plan may be.

7.3 Address Assignment

A set of rules for address assignment must be established. These rules are created in order to simplify the assignment process and help with its implementation.

7.3.1 Prefixes

In an IPv6 addressing plan it is normal to use subnet prefixes that are multiples of 4 (/48 /52 /56 /60 /64) the reason behind this is that in doing so the binary separation will always take place in between hexadecimal digits which makes it a lot easier to read and decipher the address. If the prefix used is not a multiple of 4 the binary separation will take place in the middle of one of the hexadecimal digits meaning that all the hexadecimal digits beginning with the "locked" bits will belong to this one prefix.

7.3.2 Assigned Address Block

As was mentioned earlier organizations / institutions that apply at most LIRs around the world for an IPv6 address range will be assigned with a /48 prefix. This has been the case with IPLeiria and according to that the FCCN (Federação para Computação Cientifica Nacional) has assigned the following prefix for the IPLeiria to use in its IPv6 addressing plan: **2001:690:2060::/48.**This means that we have 16 free bits to define various subnets based on locations and use types.

7.3.3 Notation of the assignable bit groups

With the amount of available IPv6 addresses we have at our disposal we can create a good amount of primary subnets. We can assign the addresses following a defined order of location and use type.

In this document we will be using the following notation for each of the assignable bits in the address. The following diagram shows the structure of the 16 bits that are assignable in order to create the various subnets that will be used in the location and use type adopted addressing plan.

Each L, U and A represents one bit of the 16 bits available for the subnet assignment. The meaning of each letter is represented bellow:

- L: location based bit
- U: use type based bit
- A: assignable bit

Table 7 depicts the assigned bit types for a Use Type based Primary Subnet network

Table 7 - Example of a Use Type Primary Subnet

2001:690:2060:	L	L	L	L	U	U	U	U	A	A	A	A	A	A	A	A	::/64

Table 8 depicts the assigned bit types for a Location based Primary Subnet network

Table 8 - Example of a Location Primary Subnet

2001:690:2060:	U	U	U	U	L	L	L	L	A	A	A	A	A	A	A	A	::/64

In both these two addressing plans we have 4 bits reserved for the location and 4 bits reserved for the use type which results in an extra 8 bits remaining that can be saved for later growth or used to further specify other useful information in the address. This allows for a good number of different locations and use types as well as providing the required space for the future growth of the network and its addressing plan.

In these specific examples we have 4 bits for the location which translates into 16 locations, 4 bits for the use type which translates to a maximum of 16 use types per location and 8 more bits for further customization, for now, these will be used to differentiate the various subnets in which the network is divided allowing for a maximum of 65536 subnets for each combination of location + use type. This leaves 64 bits free for the host side.

7.3.4 Defining the different Locations, Use Types and optional configurations

According to the Network scheme supplied to us by the IPLeiria's IT center UARS there are at least 8 different locations in the network. Using the aforementioned addressing plan methodology we have a maximum of 16 different Use Types and 16 different location possibilities which we can configure. By direct translation we can set the values of the hexadecimal digits as Table 9 shows:

Table 9 - Use Type based Addressing Plan Fields

Hexadecimal Digit	Use Type	Location
0	Students	DMZ Intranet (with server farm)
1	Teacher/Employee	DMZ Internet (with server farm)
2	Guests	C2
3	EduRoam	C5 (and local DMZ)
4	Administration	C1 (and local DMZ)
5	Unused (for future growth)	SC (and local DMZ)
6	Unused (for future growth)	Peniche (and local DMZ)
7	Unused (for future growth)	Caldas da Rainha (and local DMZ)
8 through F	Unused (for future growth)	Unused (for future growth)

Following this scheme doesn't just make it easier on the security side, like has been mentioned before by allowing stricter control over policies, but also makes it a lot easier for someone to analyze and determine the location and use type of the address they are looking at. The following example explains the procedure used to identify said characteristics:

2001:690:2060:0000::/64

The yellow highlighted digit represents the use type code for the address and the red highlighted digit represents the location code for the address. For each Use Type

code we get 8 different location codes. This happens because we decided to define a use type primary subnet instead of a location primary subnet, if the situation was the opposite then the yellow highlighted digit would represent the location and the red highlighted digit would represent the use type. The two green highlighted digits represent the extra options digits and can be used, for example to define more specific subnets within the location. For example if we have a **2001:690:2060:0200::/64** prefix and a **2001:690:2060:0400::/64** prefix, from looking at the code table, we can easily determine that the first address corresponds to a student (use type 0) who is logged in at campus 2 (location code 2) in the 00 campus 2 subnet and the second address corresponds to a student (use type 0) who is logged in at campus 1 (location code 4) in the 00 campus 1 subnet.

In regards to the two subnet hexadecimal digits both of them together account for 16bits which translates to a maximum of 65536 subnets for each use type + location combination. They can be used to specify class rooms, offices, laboratories and other more specific locations. Alternatively they can describe types of locations such as all class rooms, all IT labs, all teacher offices and so forth. For this report, and according to the needs specified to us by the IT center UARS, these digits will define the type of location. The various types of locations defined are shown by Table 10.

Table 10 - Subnet Type description

Hexadecimal Digit	Subnet Type
00	Class rooms Open Computer rooms Service rooms (reprography, student store, etc)
01	IT class rooms (including IT Labs)
02	Teacher Offices
03	Project development rooms

Therefore a Student logging in at an IT class room in Campus 2 would get assigned an address with the prefix **2001:690:2060:0201::/64**

Following this addressing plan the current IPLeiria network structure and subnet map would look like Figure 35's depiction.

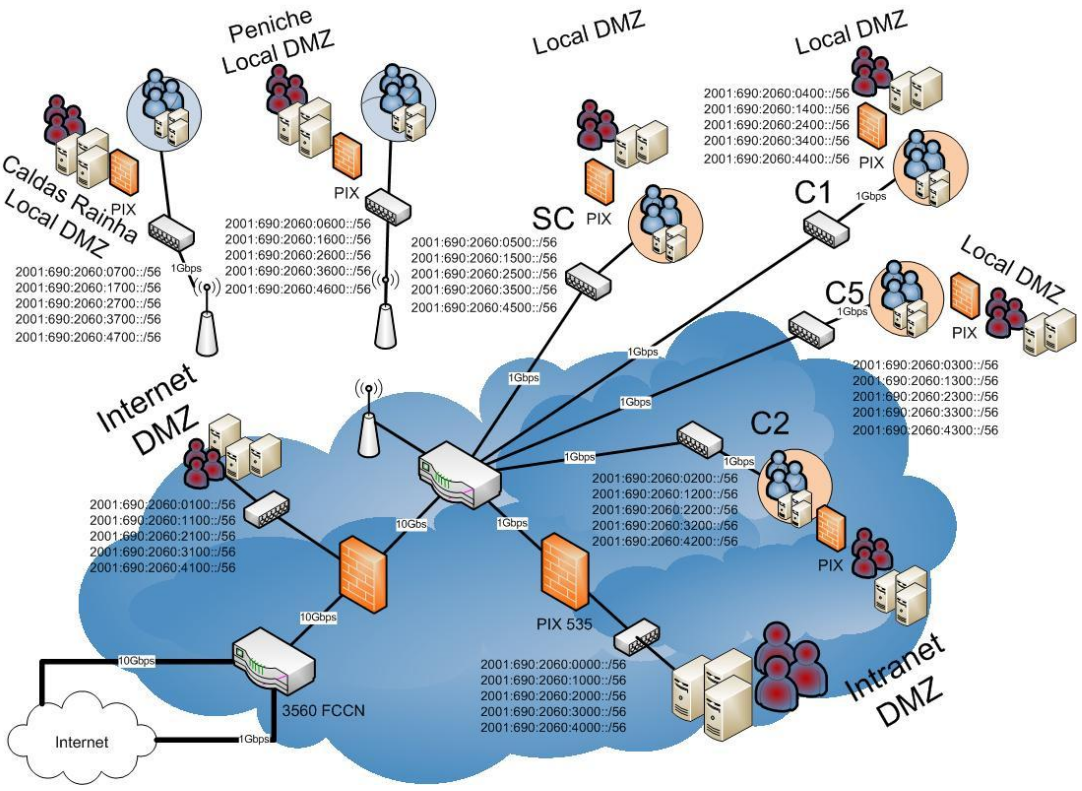


Figure 35 - IPv6 Addressing Plan

Each subnet corresponds to a unique set of use type and location parameters.

7.4 Differences between Location Primary Subnets and Use Type Primary Subnets

The main benefit to making the location the primary subnet digit is the optimization of routing tables. If we decide to assign a subnet to each building / department / etc all the networks within a single location will be aggregated to a single route in the routing table effectively compacting it and making the routing process more efficient.

If, by the other hand, it is decided to make the use type the primary subnet this routing optimization is lost and therefore more entries need to be inserted into the routing table making it larger and less efficient. This should not be a problem for the equipment though, unless the amount of entries in the routing table is really large. However the benefit gained is well worth this trouble because by taking this option we make it much easier to implement various security policies which can easily be defined based on what kind of user is logged in at the terminal since most firewall policies are based not on the location of the network but instead on the type of use.

For this reason we have decided to implement a Use Type Primary Subnet thus making it easier to control and manage the network, one very important need in the IPLeiria's network.

7.5 Alternative Addressing Plan

If a Use Type addressing plan does not fit the requirements of the network administrator and if he or she would like to analyze a possible alternative situation then the use of a location addressing plan should be taken into consideration. However the pros and cons of such a decision have already been stated, security implementation will be more complicated if this alternative is taken.

Table 11 shows an example of a possible location based addressing plan calculated for the IPLeiria's network.

Table 11 - Location Based Addressing Plan

Hexadecimal Digit	Location
0	DMZ Intranet (with server farm)
1	DMZ Internet (with server farm)
2	C2
3	C5 (and local DMZ)
4	C1 (and local DMZ)
5	SC (and local DMZ)
6	Peniche (and local DMZ)
7	Caldas da Rainha (and local DMZ)
8 through F	Unused (for future growth)

It is possible to also add a use type digit after the location type digit in this addressing plan, however if so is done then it is recommended to swap the location digit with the use type digit and use a use type addressing plan instead due to the fact that this is the easiest way to integrate IPv6 addressing with existing policies and procedures.

As is the case with use type addressing plans, a location based addressing plan can also specify more specific subnets within a location, such as a class room or IT computer room or network lab.

Table 12 defines various types of subnet types.

Table 12 - Subnet Type description

Hexadecimal Digit	Subnet Type
000	Class rooms Open Computer rooms Service rooms (reprography, student store, etc)
001	IT class rooms (including IT Labs)
002	Teacher Offices
003	Project development rooms

Therefore, if we want to announce a route to the whole CAMPUS 2 network then the aggregated announced route for that network can be 2001:690:2060:2000::/64 with 2 depicting the C2 location and 000 a classroom type of subnet.

7.6 VLANs

VLANs can be used as an added security feature. Allied with a structured addressing plan VLANs can effectively raise the level of security in an IPv6 network the same way as in IPv4.

By marking different user types with different VLAN numbers we can create security policies at the switching level by blocking access to certain ports to packets marked with a certain VLAN number. For example by marking student users with VLAN number 1 and developing a security policy that prohibits access to the server DMZ to all VLAN number 1 users it is possible to restrict access to such servers. This can be used as an added level of security allied with firewall policies and a well-structured use type addressing plan.

7.7 Point-to-Point Link Addressing

In ancient days it was usual to configure the end point of the link on the router. We would configure the remote-address or the dest_address according to the opposite end of the link, much like we do now with an IPv6 Manual tunnel. Nowadays however we use a /30 mask to create a subnet with 4 addresses and use that subnet to address both ends of the link (Figure 36). For example:

- 192.168.10.0/30 subnet for the link
- 192.168.10.0: Address of the Network
- 192.168.10.1: Router A's Address
- 192.168.10.2: Router B's Address
- 192.168.10.3: Broadcast Address of the Network

Figure 36 - Point-to-point Link

However, in IPv6 the standard for a point-to-point link connection states that a /64 prefix must be used instead of a /127 that we would instinctively use. According to the RFC 3627 [42] because the use of a /127 prefix only allows for 2 addresses within the addressing range. This results in the loss of the IPv6 subnet anycast address (:0) which may result in issues in the future since it is a blatant violation of the IPv6 standard rules and is forbidden [42] because, for example, the Duplicate Address Detection mechanism will not allow the second configured router to obtain its configured address.

Taking this into consideration one might be tempted to opt for a /126 prefix instead but this is also not a solution for the problem since the 4th address in the range will be unused and therefore able to result in a Ping-Pong loop. This consists of a situation where router A is assigned with the address 2001:690:2060::1 and router B is assigned with the address 2001:690:2060::2. If a packet is inserted into the link with a destination address of 2001:690:2060::25, for example, it will be forwarded back and forth by both routers which "think" the destination of said packet is indeed the other side of the link when in fact it just so happens the address is not reachable.

Note that this is not a problem over Ethernet because of the Layer2 address resolution performed by ARP (or ND in IPv6) which results in an ICMP host

unreachable message but for point-to-point links such as a serial connection between routers this problem does exist.

Following these issues a solution was developed and exposed in the RFC 4443 [43]. Explained in the RFC is a specific situation in which a destination unreachable message is sent with a code 3 in response to a packet received by a router from a point-to-point link, destined to an address within a subnet assigned to that same link (other than one of the receiving router's own address). In such a case, the packet MUST NOT be forwarded back onto the arrival link.

This means that if said packet was inserted into the link with its destination address of 2001:690:2060::25 it would firstly be forwarded from router A to router B but router B would drop it instead of forwarding it back to router A because of 2 things, namely the incoming interface being the same as the outgoing interface and the destination address being on that same link as the incoming interface (the one where the packet was received from).

This way it is possible to configure a /64 prefix for a point-to-point link connection without the fear of a Ping-Pong loop. This is the standard and should always be followed; there is no need to reinvent the wheel [44].

Chapter 8 VoIP

The telephone is one of the most important inventions of the 19th century and, according to some, the precursor to the information age we have today. Granted, of course, that it was not the first invention to allow for long range communications to take place, the telegraph had already been invented and was in use but the invention and development of the telephone meant that, with time, every home in the world could eventually be connected to the telephone network, effectively becoming the first real world network ever conceived.

Since its first debut during the 1870s the telephone did not stop for anything, with each passing year more and more people adopted the invention and by doing so made sure that their houses would get connected with each other. Having a conversation with a distant friend of relative was now as easy as talking to your own hand.

As the phone network developed technology kept its pace alongside it, creating better and more robust phone centrals, massive switchboards where a person's phone line gets connected to the target phone number and connection is established. This phone network architecture can be taken into account when understanding how VoIP was developed and how it works.

In the beginning of the internet, before broadband connections were developed dial up modems were used to connect a computer to the young internet. The phone line was used for data transfer and by doing so these two networks began to merge. Granted that these data and voice transfers were kept separate the physical means through which they both traveled was the same, the Plain Old Telephone Service also known as POTS was the way many of us first got to experience a connected world, a freedom to explore and share information in a way never seen before, the internet's gestation period had begun.

However, the market began to shift and what was first regarded as the supreme communication technology slowly began to get overshadowed by its little brother.

The Internet grew at such a large rate that people started to use it more for communication than the telephone service that had previously been king.

Before the 1990s there had been several proofs of concept and demonstrations given throughout the years but it was during that decade that a number of individuals integrated in different research environments, be them corporate or education institutions, began to delve into the possibility of carrying voice and video over an IP network such as a LAN or the Internet. Nowadays this technology is known as VoIP and it basically consists of the process of breaking up an audio and video transmission into tiny bits, transmitting those tiny audio and video bits through a computer network and reassembling them on the other side so that two or more people can communicate using audio and video.

And so we enter the new millennium, where phone communication is dying significantly compared to other forms of communication, where a broadband connected computer can log on to Instant Messaging networks that range from the old IRC network to the more broadly adopted Live Messenger network, with social networking sites like Twitter and Facebook, the need for phone communication has dropped significantly.

VoIP is only possible due to this growth; or rather it is only feasible due to it. With the mass adoption of broadband connections in nearly every home and office connected to the internet the amount of bandwidth available for IP Telephony is huge and an even better reason is that VoIP calls between two VoIP clients are free of charge [45].

When comparing with a POTS call a VoIP call has virtually no cost, it is effectively data, bits and nothing else, being transmitted, much like an email or an FTP transmission. Nowadays, thanks to the abrupt drop in bandwidth price most, ISPs provide connections with no download or upload traffic limit which makes data transfer costless to the end user. Compared with what phone companies offer, a VoIP solution is much more enticing and its use is a lot more desirable than a POTS solution.

Add to the fact that the use of a VoIP solution effectively forgoes the need to install a native POTS line because it uses the computer network's physical line and the cost margin gets even wider.

Another reason why a VoIP solution is very interesting is the fact that it is possible to aggregate it with the regular POTS network. It is possible to call a POTS number from a VoIP client. The data stream from the VoIP client is transformed into analog POTS signals by an analog terminal adapter, usually situated in a Telephone company's central office exchange building and transmitted through the POTS line to the called number. When the person on the POTS side answers the analog signals make a similar trip but this time around they go the other way, being transformed from analog signal to digital data and transmitted through the IP network to the VoIP client [46] .

And finally the fact that this technology can be used to transfer other types of media, most notably video is also a grand addition to why it is such a desirable technology not just for the current day and age but for the future as well. To be able to communicate through audio and video with a friend or a loved one, not just listening to their voice but also seeing their face as well. To be able to connect 10 or more people together through a video conference chat and have a business meeting with partners all around the world without needing to travel to do it. To be able to get a doctor appointment and diagnosis online if either the doctor or the patient cannot physically be at the doctor's office at the given time. Whatever the case, this is a very desirable technology that is still growing and gaining new adopters every day.

8.1 SIP VoIP architecture

The VoIP paradigm can be implemented through various methods. Some of those methods are proprietary like Skype while others are open source standards that can be used freely and implemented in a wide variety of equipments. Such is the case for the Session Initiation Protocol or SIP.

SIP is an IETF standard, it's a protocol used to establish and control VoIP communications and other data transmissions. This protocol has had a few versions over the years with its latest being specified in the RFC 3261 and is used to create, modify and terminate unicast (one peer-to-one peer) and multicast (multiple peers) connections that consist of one or more streams of media. The modification of a session consists of one or more actions including changing addresses or ports, inviting new participants or dropping ones already connected, as well as adding or removing media streams.

SIP is a simple protocol; its design is very similar to the Hypertext Transfer Protocol (HTTP) request/response transaction model. It works in conjunction with other protocols and is only involved in the signaling portion of a communication session. It works in client/server architecture and users connecting to it typically use TCP and UDP port numbers 5060 and/or 5061.

It is mainly used in setting up and tearing down voice or video calls but it has also been used in messaging applications such as event subscription and notification as well as Instant Messaging (IM). When starting a communication using a SIP provider all the joiners of said communication must agree to a set of rules, like which audio and video codec to use, before the session will begin.

8.2 IPv6 VoIP Server Specification

In this project we have setup and installed a working dual stack SIP architecture VoIP server. It is running a CentOS based linux distribution, Elastix 2.0.4 (beta 4 release date: 11th May 2011) with full IPv6 support. Installed and running is version 1.8.4.2 of Asterisk VoIP server.

Table 13 shows the configured addresses for the VoIP server that has been installed.

Table 13- VoIP server IP addresses

	Address
IPv4	172.16.20.220 (routable only from the IPv6 network's L.C.A. switch)
IPv6	2001:690:2060::CAFE

8.2.1 Server configurations

The Asterisk server has setup six different PBX Extensions: 1000, 2000, 3000, 4000, 5000 and 6000. For ease of testing all of the six extensions are configured with the same password.

In order to test IPv6 support we experimented with a range of different soft phone software and realized that only two of them seemed to work correctly as they were the only two that were able to successfully register with the server. The two VoIP clients that managed to support IPv6 connectivity are linphone-3.4.3 and jitsi-1.0-beta1-nightly.build.3555. Both of them work well in voice calls but while LinPhone did support Video Calls, Jitsi did not.

Along with the soft phones we also tested three available IP phones: Linksys SPA 922, SNOM 300 and Polycom 330. Unfortunately we came to the conclusion that neither of them supports IPv6 since all three were unable to register with the server and acquire their PBX extension registration even after a firmware update. However all three did work fine in IPv4; they were able to register, make and receive calls.

Asterisk 1.8.4.2 comes included with a fresh install of Elastix 2.0.4 (beta 4) which already includes full IPv6 SIP support. However we had to edit a few configuration files because these options don't come enabled by default. The file changes are as follows:

/etc/asterisk/sip_general_custom.conf

```
;Activate IPv6 socket
bindaddr=::

;Activate video conference
videosupport=yes
maxcallbitrate=384

;Activate codec support
allow=h261
allow=h263
allow=h263p
allow=h264
```

This file determines the SIP configuration for the server and we have activated IPv6 support, Video conference support as well as allowed video codec's.

/etc/sysconfig/network

```
NETWORKING=yes
NETWORKING_IPV6=yes
HOSTNAME=VoIPv6
IPV6_DEFAULTGW=2001:690:2060:
:1
```

This is the network configuration file where we have activated IPv6 support for the network interfaces.

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=dhcp
DHCPCLASS=
HWADDR=00:01:02:A4:34:64
IPV6ADDR=2001:690:2060::cafe/
64
IPV6INIT=yes
IPV6_AUTOCONF=no
ONBOOT=yes
DHCP_HOSTNAME=VoIPv6
```

This is the Interface eth0 configuration file which we have configured to support IPv6, we have defined a static IPv6 address for the server as well as disabled the

auto-configuration capability which would be acquired via Router Advertisement from the IPv6 server.

8.3 SIP VoIP Client / Server architecture

It is possible for an IPv4 only client to communicate with an IPv6 only client using the SIP Server as a gateway. In fact this is how this solution works. Since the VoIP server is setup using a dual stacked Network card it has both IPv4 and IPv6 connectivity, therefore it is able to communicate with IPv4 and IPv6 nodes. These nodes cannot communicate directly with each other since an IPv4 only node cannot "understand" an IPv6 packet and the same goes for an IPv6 only node; it cannot "understand" an IPv4 packet.

The Asterisk server keeps track of extensions that have been registered and their respective IP addresses, be they IPv4 or IPv6. When a node wishes to communicate with another node that is registered at the server all the traffic is routed through the server to the destination client and vice versa, when the destination client wants to send a reply to the source client it will route the reply through the VoIP server. This really makes the VoIP server resemble a plain old telephone switchboard since all communication traffic must be routed through it.

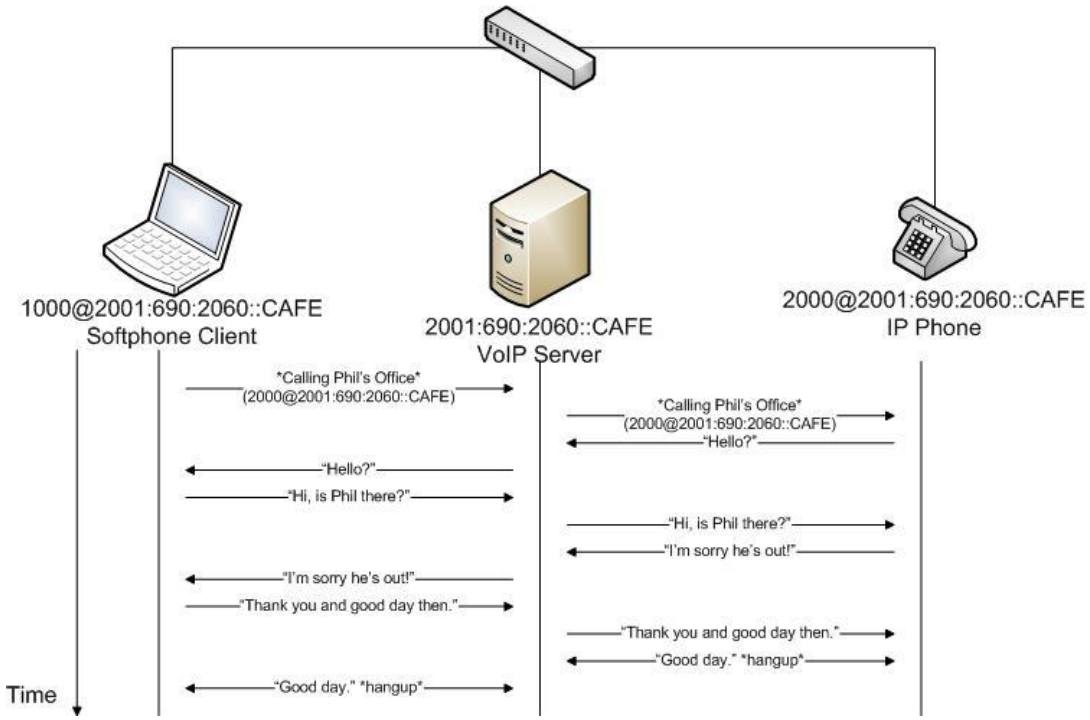


Figure 37 -VoIP Call time chart

Figure 37 shows a simple example of how the communication between two registered SIP clients works. This happens in IPv4 as well as IPv6, in the VoIP test scenario chapter we will demonstrate how an IPv6 node can call and communicate with an IPv4 node as long as they are both registered with the SIP VoIP server.

Note: We have configured an FTP server on the IPv6 server (2001:690:2060::2 / ipv6.estg.ipleiria.pt) which allows for anonymous login, we have done this so that we can provide an easy way for users connected to the IPv6 subnet to download and install the VoIP clients we have used in our VoIP work. Note that the clients provided in the FTP server are for windows OS, if using a linux distribution please download and install them through the distribution's own package manager. If you would like to try out the VoIP server simply connect to the IPv6 network, either by wireless connection or via the switch, and download the given files using an FTP client.

Chapter 9 VoIP Tests

In order to try out and test the VoIP capabilities of the server we have setup and ran a few tests. We used both IPv4 and IPv6 in these tests. We tried out three different tests, one of them with both user agents running over IPv4, another oen with both user agents running over IPv6 and a third scenario where one user agents is connected through IPv4 while the other one is connected through IPv6. This last one is without a doubt the most intriguing one opening up some very interesting possibilities.

9.1 Test 1- IPv4 only VoIP Call

This is a very simple scenario where the aim is, to begin by to capturing the registration process of a SIP user agent and afterwards to capture a whole VoIP call and understand how the two clients communicate with each other. By doing so we are able to show that all communication does indeed always pass through the server and that a client-to-client connection is in fact never established.

Figure 38 depicts the equipment used for this test scenario and its configurations.

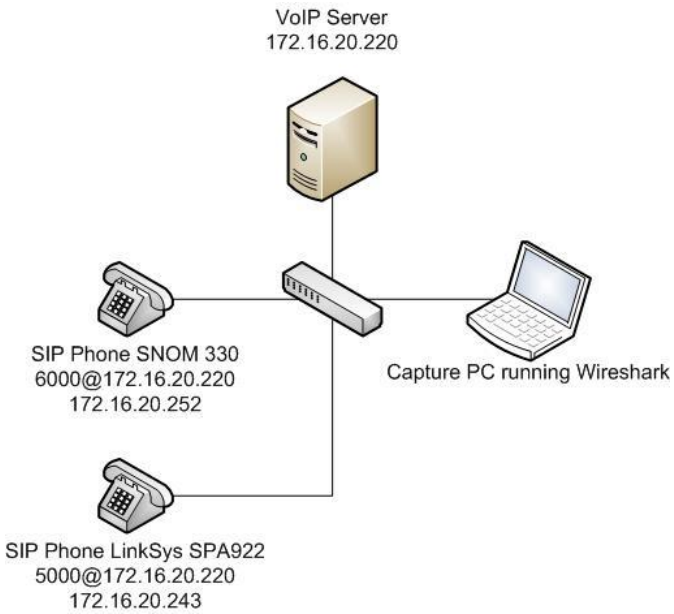


Figure 38 - VoIP Test Scenario 1

9.1.1 Step one - IPv4 LinkSys SIP Registration

With this step we aim to show the Session Initiation Protocol at work with an IPv4 SIP user agent registering with an IPv4 SIP VoIP server and the messages that both trade with each other while the registration process takes place.

	Time	Source	Destination	Protocol	Length	Info
1399	57.380595	172.16.20.243	172.16.20.220	SIP	679	Request: REGISTER sip:172.16.20.220
1400	57.381150	172.16.20.220	172.16.20.243	SIP	480	Status: 100 Trying (0 bindings)
1401	57.382237	172.16.20.220	172.16.20.243	SIP	590	Request: OPTIONS sip:5000@172.16.20.243:5060
1402	57.382460	172.16.20.220	172.16.20.243	SIP	596	Status: 200 OK (1 bindings)
1403	57.393140	172.16.20.243	172.16.20.220	SIP	455	Status: 200 OK

Figure 39 - SIP registrations (IPv6)

By looking at the captured packets (Figure 39) we can see the messages that are traded between the SIP user agent and the server while the registration process happens.

Figure 40 shows a detailed view of a captured SIP registration packet.

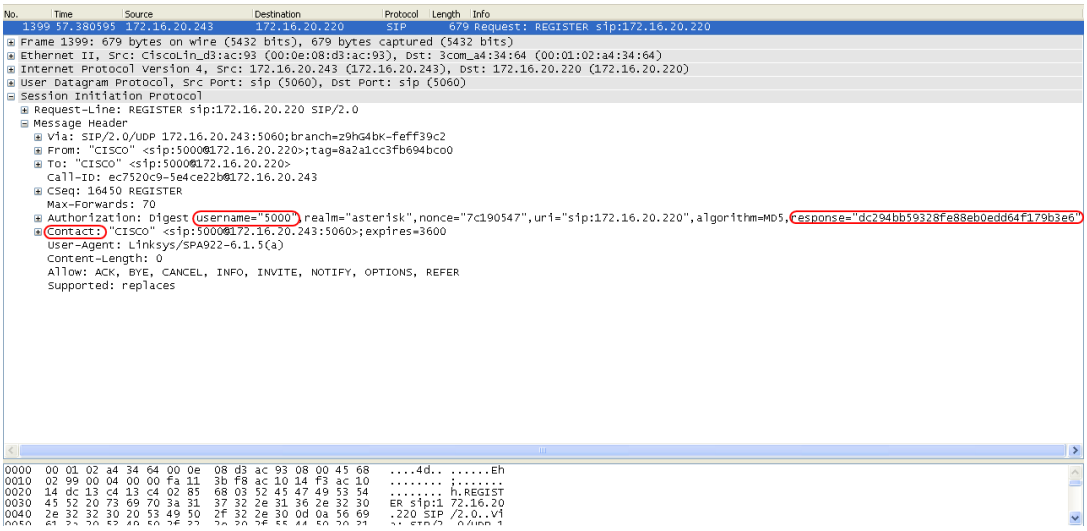


Figure 40 - SIP registration step 1 (IPv6)

The Client begins by requesting a registration with the server (packet 1399). It identifies itself with the username and requests the registration. It sends information back to the server in order to complete this step. Contact is used so the server can know how to reach the user agent that registers the Extension and the algorithm and response field are used for authentication purposes. Instead of sending the password unencrypted through the network the client instead uses it along with a bunch of more data to calculate the checksum that is sent in the "response" field. The server re-calculates that checksum and compares it with the one received if they both match the client is authenticated [47].

After authenticating and registering the server replies to the SIP User Agent with a status OK (packet 1402) and binding message to which the user agent replies with an OK message (packet 1403) and from then on the user agent is registered with the server and communication can begin between both, including calling other server registered user agents.

9.1.2 Step two - Call between two SIP user agents

Reminder:

LinkSys phone user agent (5000@172.16.20.220, IP: 172.16.20.243)

SNOM user agent (6000@172.16.20.220, IP: 172.16.20.252)

Step two consists of a SIP call between two user agents. Both are registered at the server and the call is initiated by 5000 trying to call 6000. The first thing that needs to be done in step 2 is to initiate the RTP session between the two user agents.

Figure 41 shows the packets traded during the call initiation process. In order to do so the following happens:

No.	Time	Source	Destination	Protocol	Length	Info
22	5.063598	172.16.20.243	172.16.20.220	SIP/SDP	1112	Request: INVITE sip:6000@172.16.20.220, with session description
23	5.066224	172.16.20.220	172.16.20.243	SIP	516	Status: 100 Trying
26	5.142918	172.16.20.220	172.16.20.252	SIP/SDP	998	Request: INVITE sip:6000@172.16.20.252:2049;line=yeajqfh0, with session description
27	5.143800	172.16.20.220	172.16.20.243	SIP	532	Status: 180 Ringing
28	5.178368	172.16.20.252	172.16.20.220	SIP	558	Status: 180 Ringing
29	5.180800	172.16.20.220	172.16.20.243	SIP	532	Status: 180 Ringing
30	5.686716	172.16.20.252	172.16.20.220	SIP	558	Status: 180 Ringing
33	6.190109	172.16.20.252	172.16.20.220	SIP/SDP	993	Status: 200 Ok, with session description
34	6.191020	172.16.20.220	172.16.20.252	SIP	468	Request: ACK sip:6000@172.16.20.252:2049;line=yeajqfh0
35	6.192437	172.16.20.220	172.16.20.243	SIP/SDP	820	Status: 200 OK, with session description
36	6.199240	172.16.20.243	172.16.20.220	SIP	600	Request: ACK sip:6000@172.16.20.220:5060
38	6.217886	172.16.20.252	172.16.20.220	RTP	70	Unknown RTP version 0
39	6.221310	172.16.20.243	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x935EF4BF, Seq=5140, Time=394903747
40	6.222294	172.16.20.220	172.16.20.252	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x7FECDF71, Seq=8151, Time=394903744, Mark

Figure 41 - SIP INVITE and Call initiation (IPv4)

The call is initiated by 5000 contacting the server with an INVITE request aimed at 6000 along with a description of the session parameters like audio and video codecs. (Packet 22). The server sends a TRYING response back to 5000 (packet 23) and contacts 6000 with the INVITE request from 5000 (packet 26) including the session description data. Afterwards the server informs 5000 that the call is RINGING (packet 27) and keeps forwarding RINGING messages from 6000 (packets 28, 29, 30). 6000 then answers the server with an OK response signaling the user has picked up the phone and is ready to speak (packet 33). The session description is also sent with packet 33. The server ACKnowledges 6000's OK message (packet 34) and forwards it to 5000 (packet 35) along with 6000's session description. 5000 ACKnowledges the server's message (packet 36) and starts the RTP transmission using session description that is compatible with both user agents (packets 39+).

After the Session has been started Real Time Protocol (RTP) communication begins. As described previously the user agents do not communicate directly between themselves instead using the SIP server as a Call switchboard with each packet destined to the other user agent being forwarded through the Server.

No.	Time	Source	Destination	Protocol	Length	Info
39	6.221310	172.16.20.243	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x935EF4BF, Seq=5140, Time=394903747
40	6.222294	172.16.20.220	172.16.20.252	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x7FECDF71, Seq=8151, Time=394903744, Mark
41	6.244328	172.16.20.243	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x935EF4BF, Seq=5141, Time=394903907
42	6.244505	172.16.20.220	172.16.20.252	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x7FECDF71, Seq=8152, Time=394903904
43	6.260804	172.16.20.243	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x935EF4BF, Seq=5142, Time=394904067
44	6.260960	172.16.20.220	172.16.20.252	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x7FECDF71, Seq=8153, Time=394904064
45	6.283478	172.16.20.243	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x935EF4BF, Seq=5143, Time=394904227
46	6.283583	172.16.20.220	172.16.20.252	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x7FECDF71, Seq=8154, Time=394904224
47	6.301155	172.16.20.243	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x935EF4BF, Seq=5144, Time=394904387
48	6.301364	172.16.20.220	172.16.20.252	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x7FECDF71, Seq=8155, Time=394904384
49	6.307938	172.16.20.252	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xBF7C0509, Seq=22258, Time=56323764, Mark
50	6.308034	172.16.20.220	172.16.20.243	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x2CA7B9DF, Seq=32365, Time=56323760, Mark
51	6.308505	172.16.20.252	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xBF7C0509, Seq=22259, Time=56324328
52	6.308685	172.16.20.220	172.16.20.243	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x2CA7B9DF, Seq=32366, Time=56324328
53	6.314947	172.16.20.252	172.16.20.220	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xBF7C0509, Seq=22260, Time=56324488
54	6.315096	172.16.20.220	172.16.20.243	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x2CA7B9DF, Seq=22263, Time=56324488

Figure 42 - RTP Communication (IPv6 VoIP call)

Figure 42 clearly shows this behavior as 5000 (172.16.20.243) only communicates with the server and 6000 (172.16.20.252) only communicates with the server as well.

9.2 Test 2- IPv6 only VoIP Call

This is a very simple scenario where the aim is, to begin by capturing the registration process of a SIP user agent and afterwards to capture a whole VoIP call and understand how the two clients communicate with each other. By doing so we are able to show that all communication does indeed always pass through the server and that a client-to-client connection is in fact never established.

Figure 43 depicts the equipment used for this test scenario and its configurations.

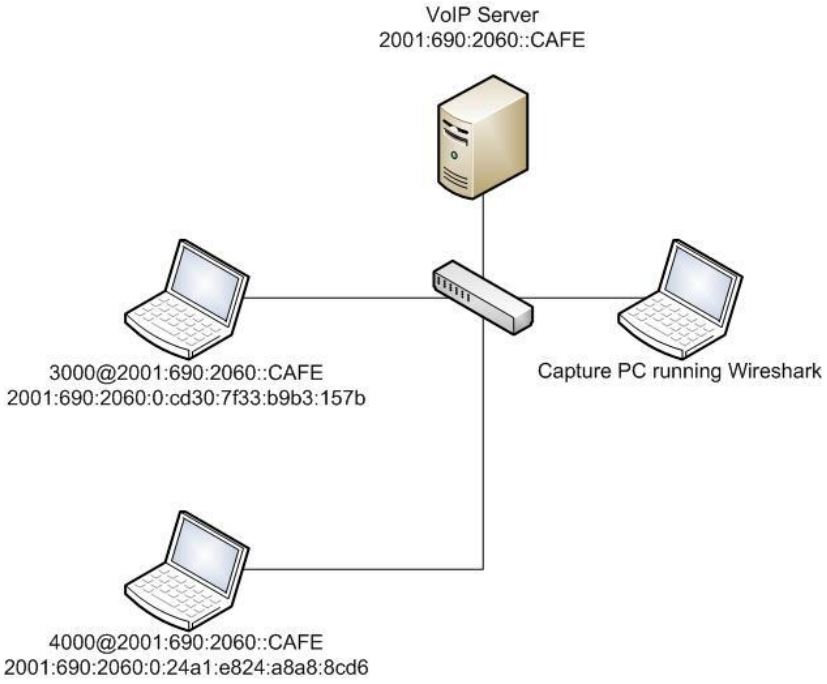


Figure 43 - VoIP Test Scenario 2

9.2.1 Step one – IPv6 Linphone (Soft Phone) SIP Registration

With this step we aim to show the Session Initiation Protocol at work with an IPv6 SIP user agent registering with an IPv6 SIP VoIP server and the messages that both trade with each other while the registration process takes place.

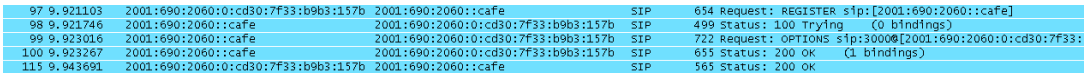


Figure 44 - SIP registration (IPv6)

By looking at the captured packets (Figure 44) we can see the messages that are traded between the SIP user agent and the server while the registration process happens.

Figure 45 shows a detailed view of a captured SIP registration packet.


```
Authorization: Digest username="3000"
response="cc386de570b46b60f6e6e7041a940475"
Contact: <sip:3000@[2001:690:2060:0:cd30:7f33:b9b3:157b]
```

Figure 45 - SIP registration step 1 (IPv6)

The Client begins by requesting a registration with the server (packet 97). It identifies itself with the username and requests the registration. It sends information back to the server in order to complete this step. Contact is used so the server can know how to reach the user agent that registers the Extension and the algorithm and response field are used for authentication purposes. Instead of sending the password unencrypted through the network the client instead uses it along with a bunch of more data to calculate the checksum that is sent in the "response" field. The server re-calculates that checksum and compares it with the one received if they both match the client is authenticated [47].

After authenticating and registering the server replies to the SIP User Agent with a status OK (packet 100) and binding message to which the user agent replies with an OK message (packet 115) and from then on the user agent is registered with the server and communication can begin between both, including calling other server registered user agents.

9.2.2 Step two - Call between two SIP user agents

Reminder:

Client 1: 3000@2001:690:2060::CAFE

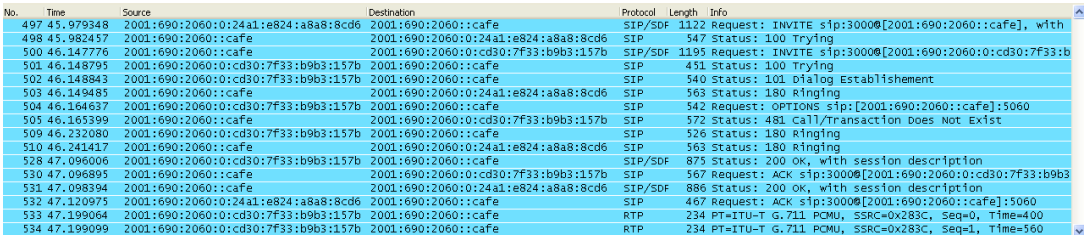
(IPv6 address: 2001:690:2060:0:cd30:7f33:b9b3:157b)

Client 2: 4000@2001:690:2060::CAFE

(IPv6 address: 2001:690:2060:0:24a1:e824:a8a8:8cd6)

Step two consists of a SIP call between two user agents. Both are registered at the server and the call is initiated by 4000 trying to call 3000. The first thing that needs to be done in step 2 is to initiate the RTP session between the two user agents.

Figure 46 shows the packets traded during the call initiation process. In order to do so the following happens:

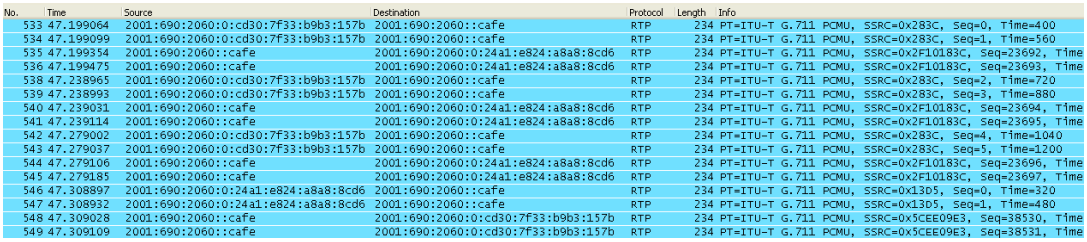


No.	Time	Source	Destination	Protocol	Length	Info
497	45.979348	2001:690:2060:0:24a1:e824:a8a8:8cd6	2001:690:2060::cafe	SIP/SDP	1122	Request: INVITE sip:3000@[2001:690:2060::cafe], with
498	45.982457	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	SIP	547	Status: 100 Trying
500	46.147776	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP/SDP	1195	Request: INVITE sip:3000@[2001:690:2060:0:cd30:7f33:b
501	46.148795	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP	451	Status: 100 Trying
502	46.148843	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP	540	Status: 101 Dialog Establishement
503	46.149485	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	SIP	563	Status: 180 Ringing
504	46.164637	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP	542	Request: OPTIONS sip:[2001:690:2060::cafe]:5060
505	46.165399	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP	572	Status: 481 Call/Transaction Does Not Exist
509	46.232080	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP	526	Status: 180 Ringing
510	46.241417	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	SIP	563	Status: 180 Ringing
528	47.096006	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP/SDP	875	Status: 200 OK, with session description
530	47.096895	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP	567	Request: ACK sip:3000@[2001:690:2060:0:cd30:7f33:b9b3
531	47.098394	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	SIP/SDP	886	Status: 200 OK, with session description
532	47.120975	2001:690:2060:0:24a1:e824:a8a8:8cd6	2001:690:2060::cafe	SIP	467	Request: ACK sip:3000@[2001:690:2060::cafe]:5060
533	47.199064	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=0, Time=400
534	47.199099	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=1, Time=560

Figure 46 - SIP INVITE and Call initiation (IPv6)

The call is initiated by 4000 contacting the server with an INVITE request aimed at 3000 along with a description of the session parameters like audio and video codecs. (Packet 497). The server sends a TRYING response back to 4000 (packet 498) and contacts 3000 with the INVITE request from 4000 (packet 500) including the session description data. Afterwards the server informs 4000 that the call is RINGING (packet 503) and keeps forwarding RINGING messages from 3000 (packets 509,510). 3000 then answers the server with a OK response signaling the user has picked up the phone and is ready to speak (packet 528). The session description is also sent with packet 528. The server ACKnowledges 3000's OK message (packet 530) and forwards it to 4000 (packet 531) along with 3000's session description. 4000 ACKnowledges the server's message (packet 532) and starts the RTP transmission using session description that is compatible with both user agents (packets 533+).

After the Session has been started Real Time Protocol (RTP) communication begins. As described previously the user agents do not communicate directly between themselves instead using the SIP server as a Call switchboard with each packet destined to the other user agent being forwarded through the Server.



No.	Time	Source	Destination	Protocol	Length	Info
533	47.199064	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=0, Time=400
534	47.199099	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=1, Time=560
535	47.199354	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x2F10183C, Seq=23692, Time
536	47.199475	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x2F10183C, Seq=23693, Time
538	47.238965	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=2, Time=720
539	47.238993	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=3, Time=880
540	47.239031	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x2F10183C, Seq=23694, Time
541	47.239114	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x2F10183C, Seq=23695, Time
542	47.279002	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=4, Time=1040
543	47.279037	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x283C, Seq=5, Time=1200
544	47.279106	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x2F10183C, Seq=23696, Time
545	47.279185	2001:690:2060::cafe	2001:690:2060:0:24a1:e824:a8a8:8cd6	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x2F10183C, Seq=23697, Time
546	47.308897	2001:690:2060:0:24a1:e824:a8a8:8cd6	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x13D5, Seq=0, Time=320
547	47.308932	2001:690:2060:0:24a1:e824:a8a8:8cd6	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x13D5, Seq=1, Time=480
548	47.309028	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x5CEE09E3, Seq=38530, Time
549	47.309109	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x5CEE09E3, Seq=38531, Time

Figure 47 - RTP Communication (IPv6 VoIP call)

Figure 47 clearly shows this behavior as 4000 (2001:690:2060:0:24a1:e824:a8a8:8cd6) only communicates with the server and 3000 (2001:690:2060:0:cd30:7f33:b9b3:157b) only communicates with the server as well.

9.3 Test 3- IPv6 / IPv4 VoIP Call

In this scenario we have setup an IPv4 only SIP VoIP client and an IPv6 only SIP VoIP client along with a dual stacked SIP VoIP server. Both clients communicate with the server using their own IP stack and, due to the fact that the server behaves as a gateway for both clients they can communicate with each other (through the server) even if they both use different IP versions.

Figure 48 depicts the equipment used for this test scenario and its configurations.

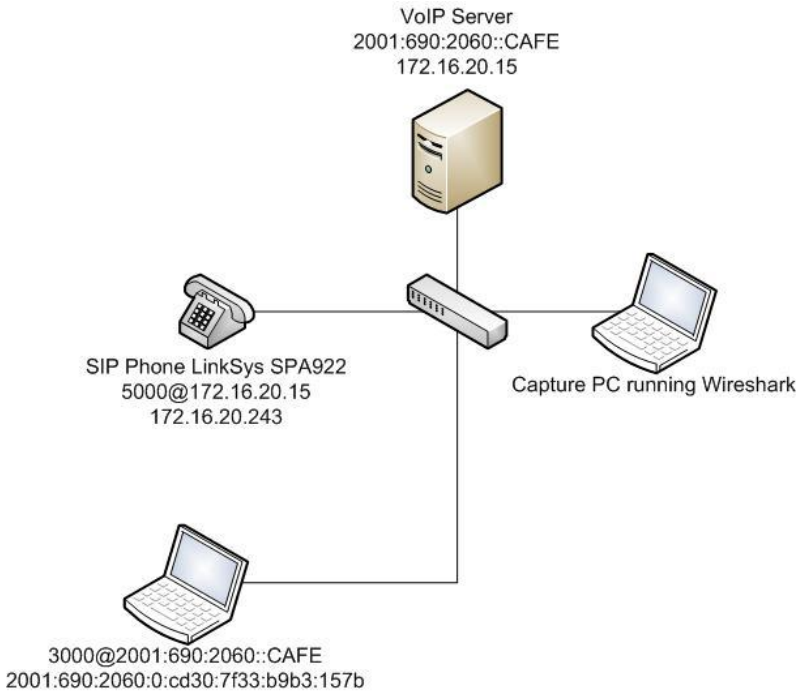


Figure 48 - VoIP Test Scenario 3

The server is able to open an IPv4 packet and extract its data field, place it in an IPv6 packet data field and inject the corresponding IPv6 header fields of the destination node allowing for IPv6 / IPv4 communication between both clients. A similar process is done when going through the same route in the opposite direction this time with the IPv6 data field being extracted and injected into an IPv4 packet and the header fields changed to reach the destination IPv4 address.

This is possible because the user agents are registered with the server and in turn the server keeps track of how to reach all its extensions, as long as the server remains configured with a dual stacked network interface this IPv4 with IPv6 communication remains possible.

9.3.1 Step one – SIP Registration

SIP registration has already been shown in the previous two scenarios and happens exactly the same way. IPv4 user agents register the same way as described in

Scenario 1 while IPv6 user agents register the same way as described in Scenario 2. Further explanation and demonstration on this process is therefore unnecessary.

9.3.2 Step two - Call between two SIP user agents

Reminder:

Client 1: 5000@172.16.20.215

(IPv4 address: 172.16.20.243)

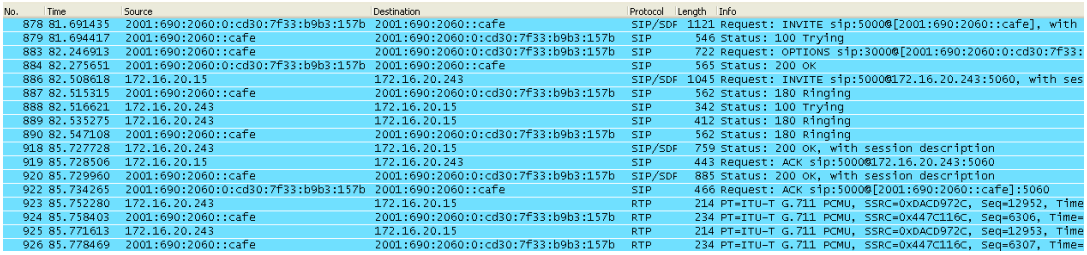
Client 2: 3000@2001:690:2060::CAFE

(IPv6 address: 2001:690:2060:0:cd30:7f33:b9b3:157b)

Step two consists of a SIP call between two user agents. Both are registered at the server and the call is initiated by 3000 trying to call 5000. At first it might seem strange but this process happens in the same way as it did in the previous 2 scenarios except for the fact that one user agent communicates in IPv4 and another user agent communicates in IPv6. Since both clients only communicate directly with the server and since the server knows how to reach both of them, even in distinct IP stacks, they can effectively join a VoIP call with each other.

The first thing that needs to be done in step 2 is initiate the RTP session between the two user agents.

Figure 49 shows the packets traded during the call initiation process. In order to do so the following happens:



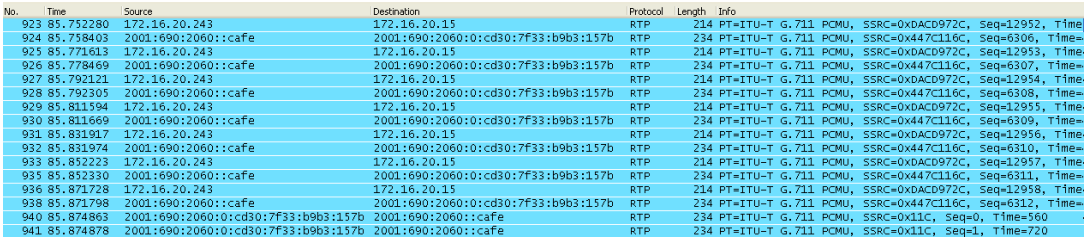
No.	Time	Source	Destination	Protocol	Length	Info
878	81.691435	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP/SDP	1121	Request: INVITE sip:5000@[2001:690:2060::cafe], with
879	81.694417	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP	546	Status: 100 Trying
883	82.246913	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP	722	Request: OPTIONS sip:3000@[2001:690:2060:0:cd30:7f33:
884	82.275651	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP	565	Status: 200 OK
886	82.508618	172.16.20.15	172.16.20.243	SIP/SDP	1045	Request: INVITE sip:5000@172.16.20.243:5060, with ses
887	82.515315	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP	562	Status: 180 Ringing
888	82.516621	172.16.20.243	172.16.20.15	SIP	342	Status: 100 Trying
889	82.535275	172.16.20.243	172.16.20.15	SIP	412	Status: 180 Ringing
890	82.547108	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP	562	Status: 180 Ringing
918	85.727728	172.16.20.243	172.16.20.15	SIP/SDP	759	Status: 200 OK, with session description
919	85.728506	172.16.20.15	172.16.20.243	SIP	443	Request: ACK sip:5000@172.16.20.243:5060
920	85.729960	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	SIP/SDP	885	Status: 200 OK, with session description
922	85.734265	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	SIP	466	Request: ACK sip:5000@[2001:690:2060::cafe]:5060
923	85.752280	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12952, Time
924	85.758403	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6306, Time=
925	85.771613	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12953, Time
926	85.778469	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6307, Time=

Figure 49 - SIP INVITE and Call initiation (IPv4 and IPv6)

The call is initiated by 3000 contacting the server with an INVITE request aimed at 5000 along with a description of the session parameters like audio and video codecs. (Packet 878). The server sends a TRYING response back to 3000 (packet 879) and contacts 5000 with the INVITE request from 3000 (packet 886) including the session description data. Afterwards the server informs 3000 that the call is RINGING (packet 887) and keeps forwarding RINGING messages from 5000 (packets 889, 890). 5000 then answers the server with an OK response signaling the user has picked up the phone and is ready to speak (packet 918). The session description is also sent with packet 918. The server ACKnowledges 5000's OK message (packet 919) and forwards it to 3000 (packet 920) along with 5000's session description. 3000 ACKnowledges the server's message (packet 922) and

starts the RTP transmission using session description that is compatible with both user agents (packets 923+).

After the Session has been started Real Time Protocol (RTP) communication begins. As described previously the user agents do not communicate directly between themselves instead using the SIP server as a Call switchboard with each packet destined to the other user agent being forwarded through the Server. However, differently from previous scenarios this feature is even more benefic; it allows IPv4 only and IPv6 only clients to communicate amongst themselves. This is a great feature for an IPv6 VoIP transition solution.



No.	Time	Source	Destination	Protocol	Length	Info
923	85.752280	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12952, Time
924	85.758403	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6306, Time=
925	85.771613	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12953, Time
926	85.778469	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6307, Time=
927	85.792121	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12954, Time
928	85.792305	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6308, Time=
929	85.811594	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12955, Time
930	85.811669	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6309, Time=
931	85.831917	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12956, Time
932	85.831974	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6310, Time=
933	85.852223	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12957, Time
935	85.852330	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6311, Time=
936	85.871728	172.16.20.243	172.16.20.15	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xDACD972C, Seq=12958, Time
938	85.871798	2001:690:2060::cafe	2001:690:2060:0:cd30:7f33:b9b3:157b	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x447C116C, Seq=6312, Time=
940	85.874863	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x11C, Seq=0, Time=560
941	85.874878	2001:690:2060:0:cd30:7f33:b9b3:157b	2001:690:2060::cafe	RTP	234	PT=ITU-T G.711 PCMU, SSRC=0x11C, Seq=1, Time=720

Figure 50 - RTP Communication (IPv4 and IPv6 VoIP call)

Figure 50 clearly shows this behavior as 5000 (172.16.20.243) only communicates with the server and 3000 (2001:690:2060:0:cd30:7f33:b9b3:157b) only communicates with the server as well.

9.4 Conclusions

After analyzing the results of all three tests it is possible to understand how useful a solution like this can be. A SIP VoIP server setup with a dual stack configuration can communicate in either IPv4 or IPv6. This is a good step to take when implementing IPv6 transition but the really useful feature is the possibility that IPv4 user agents can communicate with IPv6 user agents and vice versa.

Since the server keeps track of how to reach each registered extension and a call is always destined to the specific extension instead of the user agents IP address the server can “translate” IPv4 VoIP packets into IPv6 ones and vice versa, rerouting them to the desired user agent. This is indeed a great feature for a transition scenario.

Also worthy of note is the fact that none of the SIP phones that were made available to us were compatible with IPv6 even after a firmware update. This means that in order to adopt an IPv6 VoIP architecture in the IPLeiria’s network these phones would either need to receive a firmware update bringing them up to IPv6 compatibility or new phones would need to be purchased. Alternatively other solutions like a headset installed in a PC can be taken into consideration as an alternative to SIP phones.

Chapter 10 Conclusions

After researching and studying the available IPv6 transition mechanics for the IPLeiria's network we have come to realize that, due to the equipment used (Cisco IOS equipment and switch based network topography) the best kind of transition mechanism that could be configured within the network is a dual stack solution. If the need should ever arise for an IPv6 encapsulated in IPv4 solution the option still remains available. We would recommend such possibility, for example, for an IPv6 backup connection, an IPv6 tunneled through the native IPv4 connection to the IPLeiria's backup ISP which would act as a useful backup option if the native IPv6 connection would for some reason not be available.

While we tried to compare IPv4 with native and tunneled IPv6 we came to the conclusion that outside of a controlled environment it is very hard to determine which protocol comes out ahead in terms of bandwidth. Even with IPv4 pulling ahead on transmission rate we still can't tell for sure which protocol has better performance because we are unable to determine all the outside factors that influenced our experiment.

The IPv6 addressing plan proposed in this project follows a set of rules that not only make it easy to understand but also make it easy to identify a user type and the location of that user just by looking at the address. This can simplify the life of the network administrator greatly when the need to identify a certain address comes up. Along with those perks the proposed addressing plan also makes it possible to easily create security policies by grouping user types in the same prefix regardless of location or extra options. The use of VPNs can also be implemented to create an extra layer of security.

A dual stacked VoIP server is a great solution for communication. By allowing IPv4 and IPv6 calls and by making it possible for user agents with exclusive IP stacks to still be able to call each other this is possibly one of the most interesting transition solutions we have experimented. The need for VoIPv6 has never been higher with emerging Asian countries just now starting to embrace the Internet. The

huge breathing space that IPv6 brings to the addressing range problematic is unmatched and the possibility to call an IPv4 only user agent from an IPv6 only user agent is amazingly welcomed.

Our participation on the World IPv6 day was very interesting and we were able to take in a broader notion of the worldwide IPv6 adoption with world renowned sites like Facebook, Google and YouTube providing their services over IPv6. We were very glad and excited to provide the school and the whole IPLeiria institution with worldwide visibility and renown. It was very fulfilling to look at the list of IPv6 enabled websites in the world IPv6 day website and see that the IPLeiria was the only Portuguese education institution in there.

The world is clearly adopting IPv6, either by curiosity and interest or simply because the IPv4 addressing range is now exhausted and soon there will be no more available IPv4 addresses. It's time to face things head on and accept it the future is coming, IPv6 is here to stay!

10.1 World IPv6 day

As mentioned before a contribution to the World IPv6 day was made with the opening of the IPv6 WAN connection during which time students and teachers had full access to try out the IPLeiria's IPv6 connection and were allowed access to various sites worldwide that provided their services over IPv6. Access to the IPLeiria's IPv6 web page was also monitored equalizing a total of 189 visits over IPv6 and a total of 1139 visits over IPv4; results that were satisfying and allowed us to take notice of the amount of people that got interested in the initiative.

10.2 Future Work

Concerning future work that should arise from this project we feel it would be interesting for students to participate in the implementation of IPv6 in the IPLeiria's network either following the Addressing Plan that we have suggested or adopting a different set of rules if the need for it arises.

Another interesting Project might be to study the inversed process, IPv4 encapsulated in IPv6. Common knowledge tells us that sooner or later the number of IPv6 only nodes will be larger than the number of IPv4 only nodes which will bring forth the need to use IPv4 encapsulated in IPv6 to connect possible IPv4 islands with each other.

A wider implementation of a dual stacked IPv6 SIP VoIP solution would also be an interesting topic. Experimenting with delivering an IPv6 SIP VoIP service to key locations within the IPLeiria's network like teacher's offices and some

administrative locations in order to experiment and test its scalability would be a very interesting work to develop.

Since neither of the SIP Phones that were made available to us by the IT center UARS had IPv6 compatibility another possible thing to look into is the testing of other, newer, phones that might be compatible with IPv6 SIP VoIP.

References

[1] California, Information Sciences Institute University of Southern, *INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, RFC: 791,* Sptember 1981.

[2] K. Das, “IPv6 - The History and Timeline,” [Online]. Available: http://ipv6.com/articles/general/timeline-of-ipv6.htm. [Accessed May 2011].

[3] xibl, 5 August 2010. [Online]. Available: http://www.xibl.com/general-articles/ipv4-vs-ipv6/. [Accessed May 2011].

[4] S. Deering e R. Hinden, *Internet Protocol, Version 6 (IPv6), RFC: 2460,* December 1998.

[5] J. Rajahalme, A. Conta, B. Carpenter e B. Carpenter, *IPv6 Flow Label Specification, RFC: 3697,* March 2004.

[6] M. Blanchet, “Migrating to IPv6,” em *Migrating to IPv6*, Québec, Canada, John Wiley & Sons, Ltd, 2006.

[7] [Online]. Available: http://2.bp.blogspot.com/_sVg1XPTI2fY/TOzsDY8kr_I/AAAAAAAAABo/ENduE2VLZ58/s1600/ipv4_ipv6.jpg. [Accessed May 2011].

[8] C. Perkins, D. Johnson e J. Arkko, *Mobility Support in IPv6, RFC: 6275,* June 2004.

[9] E. N. D. Almeida e O. A. D. S. Ines, “INTRODUÇÃO AO PROTOCOLO IPV6 E ANÁLISE DE DESEMPENHO DO IPSEC SOBRE OS PROTOCOLOS IPV4 E IPV6,” 2009. [Online]. Available: http://pt.scribd.com/doc/24420672/Introducao-ao-IPv6-e-Desempenho-do-IPSec-com-IPv4-e-IPv6. [Accessed May 2011].

[10] [Online]. Available: http://media.techtarget.com/digitalguide/images/Misc/mobile-ip-ch10-1.gif. [Accessed May 2011].

[11] N. János Mohácsi, “Kopaonik, S erbia & Montenegro,” 2006. [Online]. Available: http://www.6diss.org/workshops/see-1/qos.pdf. [Accessed May 2011].

[12] Secureit-net International Pte Ldt, “Quality of Service (QOS) in IPv6,” [Online]. Available: http://www.netdummy.net/qos.html. [Accessed May 2011].

[13] S. Frankel e S. Krishnan, *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap, RFC 6071,* February 2011.

[14] W. M. Júnior, “IPv6: A Nova Geração de Comunicação,” 02 September 2008. [Online]. Available: http://www.ipv6.br/IPV6/ArtigoNovaGeracaoComunicacaoParte07. [Accessed May 2011].

[15] T. Hain, *Architectural Implications of NAT, RFC 2993,* November 2000.

[16] S. Kent, *IP Authentication Header, RFC:4302,* December 2005.

[17] S. Kent, *IP Encapsulating Security Payload (ESP), RFC: 4303,* December 2005.

[18] E. C. Kaufman, *Internet Key Exchange (IKEv2) Protocol, RFC 4306,* December 2005.

[19] C. Kaufman, P. Hoffman, Y. Nir e P. Eronen, *Internet Key Exchange Protocol Version 2 (IKEv2), RFC 5996,* September 2010.

[20] A. J. S. Silva e R. C. Teixeira, “Arquitetura IP Security - Parte 1,” 2004. [Online]. Available: http://www.rnp.br/newsgen/9907/ipsec3.html. [Accessed May 2011].

[21] K. Das, “IPSec & IPv6 - Securing the NextGen Internet,” [Online]. Available: http://ipv6.com/articles/security/IPsec.htm. [Accessed May 2011].

[22] R. Hinden, S. Deering e E. Nordmark, *IPv6 Global Unicast Address Format, RFC 3587,* August 2003.

[23] microsoft, “Routing IPv6 Traffic over an IPv4 Infrastructure,” 28 March 2003. [Online]. Available: technet.microsoft.com/en-us/library/cc756770%28WS.10%29.aspx. [Accessed May 2011].

[24] Microsoft, “IPv6 Address Types: TCP/IP,” 28 March 2003. [Online]. Available: http://technet.microsoft.com/en-us/library/cc757359%28WS.10%29.aspx. [Accessed May 2011].

[25] C. Huitema e B. Carpenter, *Deprecating Site Local Addresses, RFC: 3879,* September 2004.

[26] Cisco, “Cisco IOS IPv6 Multicast Introduction,” [Online]. Available: http://www.cisco.com/en/US/tech/tk828/technologies_white_paper09186a0080203e90.shtml. [Accessed May 2011].

[27] E. Nordmark e R. Gilligan, *Basic Transition Mechanisms for IPv6 Hosts and Routers, RFC 4213,* October 2005.

[28] C. Sellers, “www.rmv6tf.org,” 21 April 2009. [Online]. Available: http://www.rmv6tf.org/2009-IPv6-Summit-Presentations/Chuck%20Sellers%20-%20090421-IPv6-Transition-Mechanisms-Sellers.pdf. [Accessed May 2011].

[29] R. Gilligan e E. Nordmark, *Transition Mechanisms for IPv6 Hosts and Routers, RFC 2893,* August 2000.

[30] A. Conta e S. Deering, *Generic Packet Tunneling in IPv6 Specification, RFC 2473,* December 1998.

[31] A. Durand, P. Fasano, I. Guardini e D. Lento, *IPv6 Tunnel Broker, RFC 3053,* January 2001.

[32] C. Huitema, *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs), RFC 4380,* February 2006.

[33] B. Carpenter e K. Moore, *Connection of IPv6 Domains via IPv4 Clouds, RFC 3056,* February 2001.

[34] B. Carpenter e C. Jung, *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels, RFC 2529,* March 1999.

[35] P. Grossetete, “www.ipv6-tf.com.pt,” 2001. [Online]. Available: http://www.ipv6-tf.com.pt/documentos/geral/cisco/ipv6_IntegrationAndTransition_Abr2003.pdf. [Accessed May 2011].

[36] F. Templin, T. Gleeson, M. Talwar e D. Thaler, *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP), RFC 4214,* October 2005.

[37] F. Templin, T. Gleeson e D. Thaler, *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP), RFC 5214,* March 2008.

[38] C. Aoun e E. Davies, *Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status, RFC 4966,* July 2007.

[39] F. Gont, “IPv6 security issues: IPv6 transition mechanisms,” [Online]. Available: http://searchsecurity.techtarget.com/tip/IPv6-security-issues-IPv6-transition-mechanisms?asrc=EM_NLT_13928830&track=NL-422&ad=833954&. [Accessed June 2011].

[40] wireshark, “7.8. Checksums,” [Online]. Available: http://www.wireshark.org/docs/wsug_html_chunked/ChAdvChecksums.html. [Accessed July 2011].

[41] Surf Net, “Preparing an IPv6 Addressing Plan Manual,” December 2010: Original text, March 2011: Translation provided by RIPE NCC. [Online]. Available: http://www.ripe.net/training/material/IPv6-for-LIRs-Training-Course/IPv6_addr_plan4.pdf.

[42] P. Savola, *Use of /127 Prefix Length Between Routers Considered Harmful, RFC 3627,* September 2003.

[43] A. Conta, S. Deering, M. Gupta e Ed., *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, RFC 4443,* March 2006.

[44] M. ‘. Yoshinobu, Internet Initiative Japan Inc., 2008. [Online]. Available: http://archive.apnic.net/meetings/26/program/apops/matsuzaki-ipv6-p2p.pdf. [Accessed June 2011].

[45] Packetizer, Inc., “Introduction to VoIP,” 2011. [Online]. Available: http://www.packetizer.com/ipmc/papers/understanding_voip/voip_introduction.html. [Accessed June 2011].

[46] MetaFilter Network Inc., “How does VoIP connect to POTS lines?,” 11 July 2006. [Online]. Available: http://ask.metafilter.com/41973/How-does-VoIP-connect-to-POTS-lines. [Accessed June 2011].

[47] “SIP Authentication,” 31 October 2004. [Online]. Available: http://www.voip-info.org/wiki/view/SIP+Authentication. [Accessed July 2011].

Appendix

In this appendix we provide the running config files from the IPv6 Tunneling Tests as well as the configurations for the configured dual stacked FTP server. We also give a small mention to the quality of the machine in which we installed the VoIP server.

1 Tunnel Scenarios configurations

This appendix contains router configurations for the three types of tunnels that were tested during the development of this project.

1.1 Manual Tunnel

The configuration of the manual tunnel scenario was done using 3 routers. The image below depicts this scenario:

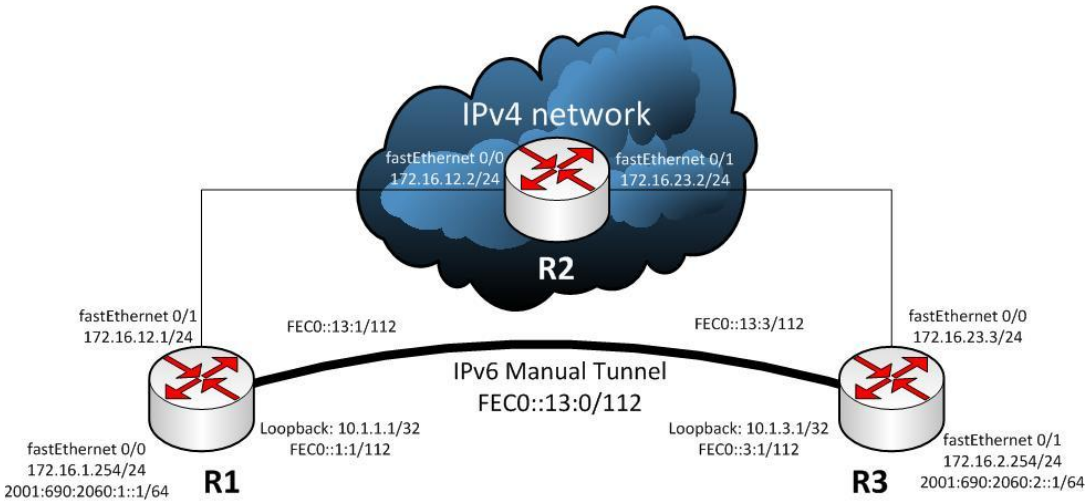


Figure 51- configuration of manual tunnels scheme

Router 1

running-config of R1.

```
version 12.4
!
hostname R1
!
ipv6 unicast-routing
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
 ipv6 address FEC0::1:1/112
 ipv6 ospf 1 area 0
!
interface Tunnel0
 no ip address
 ipv6 address FEC0::13:1/112
 ipv6 ospf 1 area 0
 tunnel source FastEthernet0/1
 tunnel destination 172.16.23.3
 tunnel mode ipv6ip
!
interface FastEthernet0/0
 ip address 172.16.1.254 255.255.255.0
 duplex auto
 speed auto
 ipv6 address 2001:690:2060:1::1/64
 ipv6 ospf 1 area 0
!
interface FastEthernet0/1
 ip address 172.16.12.1 255.255.255.0
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 10.1.3.1 255.255.255.255 172.16.12.2
ip route 172.16.2.0 255.255.255.0 172.16.12.2
ip route 172.16.23.0 255.255.255.0 172.16.12.2
!
ipv6 route 2001:690:2060:2::/64 FEC0::13:3
ipv6 router ospf 1
 log-adjacency-changes
!
end
```

Router 2

running-config of R2.

```
version 12.4
!
hostname R2
!
interface FastEthernet0/0
 ip address 172.16.12.2 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 172.16.23.2 255.255.255.0
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 10.1.1.1 255.255.255.255 172.16.12.1
ip route 10.1.3.1 255.255.255.255 172.16.23.3
ip route 172.16.1.0 255.255.255.0 172.16.12.1
ip route 172.16.2.0 255.255.255.0 172.16.23.3
!
end
```

Router 3

running-config of R3.

```
version 12.4
!
hostname R3
!
ipv6 unicast-routing
!
interface Loopback0
 ip address 10.1.3.1 255.255.255.255
 ipv6 address FEC0::3:1/112
 ipv6 ospf 1 area 0
!
interface Tunnel0
 no ip address
 ipv6 address FEC0::13:3/112
 ipv6 ospf 1 area 0
 tunnel source FastEthernet0/0
 tunnel destination 172.16.12.1
 tunnel mode ipv6ip
!
interface FastEthernet0/0
 ip address 172.16.23.3 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 172.16.2.254 255.255.255.0
 duplex auto
 speed auto
 ipv6 address 2001:690:2060:2::1/64
 ipv6 ospf 1 area 0
!
ip forward-protocol nd
ip route 10.1.1.1 255.255.255.255 172.16.23.2
ip route 172.16.1.0 255.255.255.0 172.16.23.2
ip route 172.16.12.0 255.255.255.0 172.16.23.2
!
no ip http server
no ip http secure-server
!
ipv6 route 2001:690:2060:1::/64 FEC0::13
ipv6 router ospf 1
 log-adjacency-changes
!
end
```

1.2 6to4 Tunnel

The configuration of the 6to4 tunnel scenario was done using 3 routers. The image below depicts this scenario.

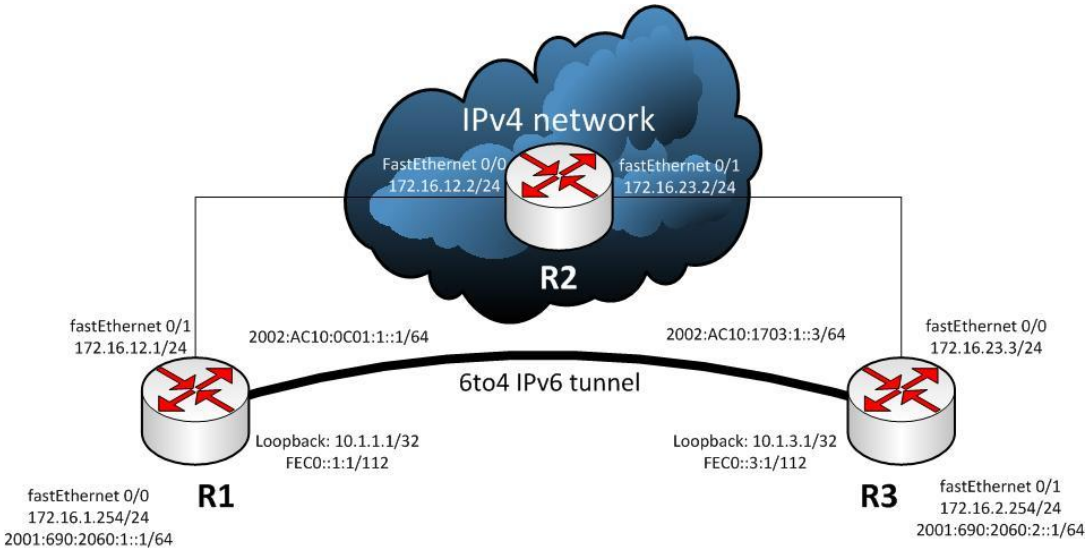


Figure 52 - configuration of 6to4 tunnels scheme

Router 1

running-config of R1.

```
version 12.4
!
hostname R1
!
ipv6 unicast-routing
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
 ipv6 address FEC0::1:1/112
!
interface Tunnel0
 no ip address
 no ip redirects
 ipv6 address 2002:AC10:C01:1::1/64
 tunnel source FastEthernet0/1
 tunnel mode ipv6ip 6to4
!
interface FastEthernet0/0
 ip address 172.16.1.254 255.255.255.0
 duplex auto
 speed auto
 ipv6 address 2001:690:2060:1::1/64
!
interface FastEthernet0/1
 ip address 172.16.12.1 255.255.255.0
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 10.1.3.1 255.255.255.255 172.16.12.2
ip route 172.16.2.0 255.255.255.0 172.16.12.2
ip route 172.16.23.0 255.255.255.0 172.16.12.2
!
ipv6 route 2001:690:2060:2::/64 2002:AC10:1703:1::3
ipv6 route 2002::/16 Tunnel0
ipv6 route FEC0::3:0/112 2002:AC10:1703:1::3
!
end
```

Router 2

running-config of R2.

```
version 12.4
!
hostname R2
!
interface FastEthernet0/0
 ip address 172.16.12.2 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 172.16.23.2 255.255.255.0
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 10.1.1.1 255.255.255.255 172.16.12.1
ip route 10.1.3.1 255.255.255.255 172.16.23.3
ip route 172.16.1.0 255.255.255.0 172.16.12.1
ip route 172.16.2.0 255.255.255.0 172.16.23.3
!
end
```

Router 3

running-config of R3.

```
version 12.4
!
hostname R3
!
ipv6 unicast-routing
!
interface Loopback0
 ip address 10.1.3.1 255.255.255.255
 ipv6 address FEC0::3:1/112
!
interface Tunnel0
 no ip address
 no ip redirects
 ipv6 address 2002:AC10:1703:1::3/64
 tunnel source FastEthernet0/0
 tunnel mode ipv6ip 6to4
!
interface FastEthernet0/0
 ip address 172.16.23.3 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 172.16.2.254 255.255.255.0
 duplex auto
 speed auto
 ipv6 address 2001:690:2060:2::1/64
!
ip forward-protocol nd
ip route 10.1.1.1 255.255.255.255 172.16.23.2
ip route 172.16.1.0 255.255.255.0 172.16.23.2
ip route 172.16.12.0 255.255.255.0 172.16.23.2
!
ipv6 route 2001:690:2060:1::/64 2002:AC10:C01:1::1
ipv6 route 2002::/16 Tunnel0
ipv6 route FEC0::1:0/112 2002:AC10:C01:1::1
!
end
```

1.3 ISATAP Tunnel

The configuration of the ISATAP tunnel scenario was done using 3 routers. The image below depicts this scenario.

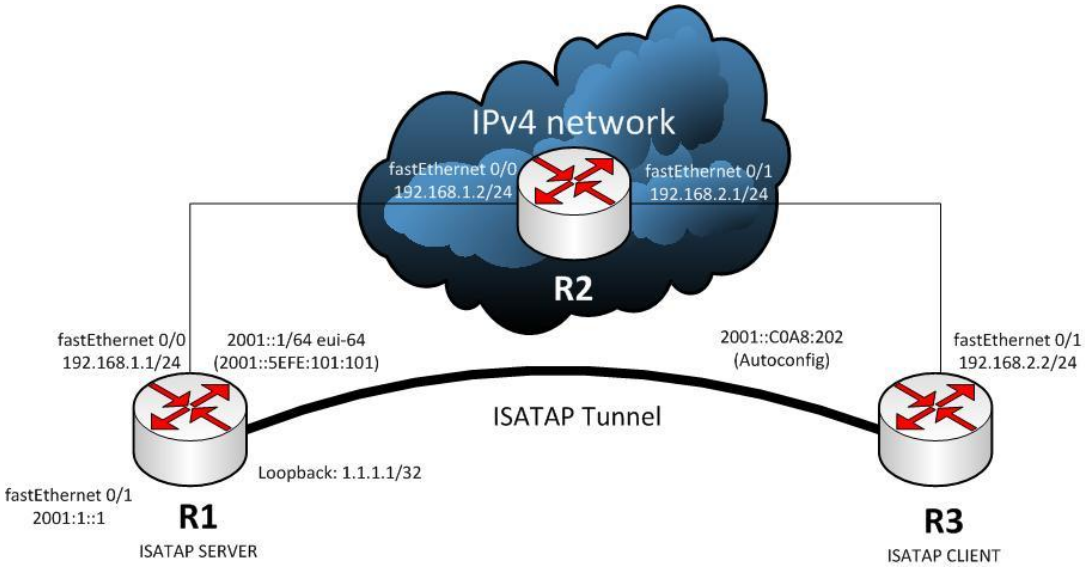


Figure 53 - scheme configuration of ISATAP tunnels

Router 1

running-config of R1.

```
version 12.4
!
hostname R1
!
ipv6 unicast-routing
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface Tunnel0
 no ip address
 no ip redirects
 ipv6 address 2001::/64 eui-64
 no ipv6 nd suppress-ra
 tunnel source Loopback0
 tunnel mode ipv6ip isatap
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 192.168.2.0 255.255.255.0 192.168.1.2
!
end
```

Router 2

running-config of R2.

```
version 12.4
!
hostname R2
!
interface FastEthernet0/0
 ip address 192.168.1.2 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.2.1 255.255.255.0
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 1.1.1.1 255.255.255.255 192.168.1.1
!
end
```

Router 3

running-config of R3.

```
version 12.4
!
hostname R3
!
ipv6 unicast-routing
!
interface Tunnel0
 no ip address
 ipv6 address autoconfig
 ipv6 enable
 tunnel source FastEthernet0/1
 tunnel destination 1.1.1.1
 tunnel mode ipv6ip
!
interface FastEthernet0/1
 ip address 192.168.2.2 255.255.255.0
 duplex auto
 speed auto
!
ip forward-protocol nd
ip route 1.1.1.1 255.255.255.255 192.168.2.1
ip route 192.168.1.0 255.255.255.0 192.168.2.1
!
end
```

2 FTP configuration

Installation and configuration of an IPv6 enabled FTP server using CentOS 5.6.

2.1 FTP install

First step is to install the packaged vsftpd

```
Yum install vsftpd.i386
```

Second step is to give the folder permissions Pub

```
Chmod 777 /var/ftp/pub
```

Third step is to change FTP file configurations. In path /etc/vsftpd/vsftdd.conf

- Enable “anonymous_enable=YES”
- Disable “listen=YES”
- Enable “listen_ipv6=YES”

2.2 Firewall configuration

2.2.1 IP6Tables

Change ip6tables. In path /etc/sysconfig/ip6tables. And add is lines to file configuration.

```
-A INPUT -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -m tcp -p tcp --dport 21 -j ACCEPT
-A RH-Firewall-1-INPUT -m tcp -p tcp --dport 22 -j ACCEPT
```

Change iptables. In path /etc/sysconfig/iptables. And add is lines to file configuration.

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
```

2.3 FTP file configurations

```
# Example config file /etc/vsftpd/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
#ATENCAO
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
#ATENCAO
#local_enable=YES
#local_enable=NO
#
# Uncomment this to enable any form of FTP write command.
#ATENCAO
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
#
# Uncomment this if you want the anonymous FTP user to be able to create
# new directories.
#anon_mkdir_write_enable=YES
#
# Activate directory messages - messages given to remote users when they
# go into a certain directory.
dirmessage_enable=YES
#
# The target log file can be vsftpd_log_file or xferlog_file.
# This depends on setting xferlog_std_format parameter
xferlog_enable=YES
```

```
#
# Make sure PORT transfer connections originate from port 20 (ftp-data).
connect_from_port_20=YES
#
# If you want, you can arrange for uploaded anonymous files to be owned by
# a different user. Note! Using "root" for uploaded files is not
# recommended!
#chown_uploads=YES
#chown_username=whoever
#
# The name of log file when xferlog_enable=YES and xferlog_std_format=YES
# WARNING - changing this filename affects /etc/logrotate.d/vsftpd.log
#xferlog_file=/var/log/xferlog
#
# Switches between logging into vsftpd_log_file and xferlog_file files.
# NO writes to vsftpd_log_file, YES to xferlog_file
xferlog_std_format=NO
#
# You may change the default value for timing out an idle session.
#idle_session_timeout=600
#
# You may change the default value for timing out a data connection.
#data_connection_timeout=120
#
# It is recommended that you define on your system a unique user which the
# ftp server can use as a totally isolated and unprivileged user.
#nopriv_user=ftpsecure
#
# Enable this and the server will recognise asynchronous ABOR requests. Not
# recommended for security (the code is non-trivial). Not enabling it,
# however, may confuse older FTP clients.
#ATENCAO
async_abor_enable=YES
#
# By default the server will pretend to allow ASCII mode but in fact ignore
# the request. Turn on the below options to have the server actually do ASCII
# mangling on files when in ASCII mode.
# Beware that on some FTP servers, ASCII support allows a denial of service
# attack (DoS) via the command "SIZE /big/file" in ASCII mode. vsftpd
# predicted this attack and has always been safe, reporting the size of the
# raw file.
```

```
# ASCII mangling is a horrible feature of the protocol.
#ascii_upload_enable=YES
#ascii_download_enable=YES
#
# You may fully customise the login banner string:
ftpd_banner=FTP ipv6.estg.ipleiria.pt
#
# You may specify a file of disallowed anonymous e-mail addresses.
Apparently
# useful for combatting certain DoS attacks.
#deny_email_enable=YES
# (default follows)
#banned_email_file=/etc/vsftpd/banned_emails
#
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().
#chroot_list_enable=YES
#chroot_list_enable=NO
# (default follows)
#chroot_list_file=/etc/vsftpd/chroot_list
chroot_list_file=/etc/vsftpd/user_list
#
# You may activate the "-R" option to the builtin ls. This is disabled by
# default to avoid remote users being able to cause excessive I/O on
large
# sites. However, some broken FTP clients such as "ncftp" and "mirror"
assume
# the presence of the "-R" option, so there is a strong case for enabling
it.
#ls_recurse_enable=YES
#
# When "listen" directive is enabled, vsftpd runs in standalone mode
and
# listens on IPv4 sockets. This directive cannot be used in conjunction
# with the listen_ipv6 directive.
#listen=YES
```

```
# This directive enables listening on IPv6 sockets. To listen on
IPv4 and IPv6
# sockets, you must run two copies of vsftpd whith two
configuration files.
# Make sure, that one of the listen options is commented !!
listen_ipv6=YES ##Alteração para escutar em IPv6 deixando o
comentado listen=yes

pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES

chroot_local_user=YES
log_ftp_protocol=YES
banner_file=/etc/vsftpd/issue
```

3 VoIP Server

The machine where we have installed the VoIP server and in which it is running does not fit the requirements of Elastix and Asterisk. It should be ported to a much more muscled computer with more memory and a stronger processor. It has been known to crash after a few days because of these issues and might need to be rebooted for testing.