instituto politécnico de leiria
escola superior de tecnologia e gestão

IPv6@IPLeiria

IPv6 Deployment in Wireless Networks

Bruno Amaral Silvestre

Carlos Alexandre Ferreira da Silva

July 2011

Computer Engineering - Final Project Course

Work conducted under the guidance of Professor Nuno Veiga

Acknowledgments

We would like to express our gratitude to our academic and research advisor Professor Nuno Veiga, for his guidance and constant support in helping us conduct and complete this work.

We would also like to thank Professors Mário Antunes, Leonel Santos and Paulo Valente for their inspiration and guidance into the field of computer networks.

Special thanks also to Eng. Adaíl Oliveira and to IPLeiria UARS (Network Management and Security Team).

Thanks to Paul Jeong and Luc Beloeil, authors of RFC 6106, and Pekka Savola, author of *radvd*, who were very forthcoming in our contacts.

Thanks to our colleagues Roberto Leal and Tiago Almeida, with whom we worked throughout this semester in development of our respective projects.

Our sincere appreciation to our families and relatives who have supported and encouraged us over the years.

(Carlos) On a personal note, a very special thanks to my girlfriend for her support throughout these times of absence.

Abstract

Is the world ready for IPv6? Earlier this year the entity responsible for assigning addresses, assigned the last available blocks, thus marking the end of available addresses under the current IPv4 system. Even though IPv6 has been around for almost two decades, few real world implementations have been considered.

This work aims to demonstrate current IPv6 readiness as it relates to Wireless solutions, and its compatibility with current well-known Operating Systems. We will investigate current solutions available, and how they can be used in integration of IPLeiria's transition effort.

So throughout this work, a thorough study of this new version of the Internet Protocol has been done, and its advantages and problems were addressed, with a careful consideration that both protocols will co-exist for years, and that sometimes using the strengths of both protocols simultaneously can be the best measure. Stateless and stateful configuration of hosts, authentication and security in the wireless medium was subject of testing and analysis.

Index

Figure Index

Table Index

Acronym List

AD	Active Directory
AH	Authentication Header
ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency Network
BA	Binding Acknowledgment
BU	Binding Update
CIDR	Classless-Inter-Domain-Routing
CPU	Central Process Unit
DAD	Duplicate Address Duplication
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
DOS	Denial Of Service
EAP	Extensible Authentication Protocol
EAPOL	Extensible Authentication Protocol Over LAN
EH	Extension Header
ESP	Encrypted Security Payload
EUI	Extended Unique Identifier
FTP	File Transfer Protocol
GPL	GNU Public License
ICMPv6	Internet Control Message Protocol Version 6
IEEE	Institute of Electrical and Electronics Engineers
IEFT	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IHL	Internet Header Length
IKE	Internet Key Exchange
Interface ID	Interface Identifier
IOS	Internetwork Operating System
IP	Internet Protocol
IPng	Internet Protocol Next Generation
IPsec	Internet Protocol Security
IPv6	Internet Protocol version 6
LAN	Local Access Network
MAC	Media Access Control
MS	Microsoft
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NCC	Network Coordinate Center
NCP	Network Control Program
ND	Neighbor Discovery
NDP	Neighbor Discovery Protocol
NPS	Network Policy Server
NTLM	NT LAN Manager
NS	Neighbor Solicitation
NTP	Network Time Protocol
OS	Operative System
OSI	Open Systems Interconnection

PAE	Port Authentication Entity
RA	Router Advertisement
RADIUS	Remote Authentication Dial In User Service
RADVD	Router Advertisement Daemon
RDNSS	Recursive Domain Name System Server
RIR	Regional Internet Registries
RFC	Request For Comment
RHEL	Red Hat Enterprise Linux
RS	Router Solicitation
SA	Security Association
SIP	Session Initiation Protocol
SLAAC	Stateless Address Autoconfiguration
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office
SPI	Security Parameter Index
SSH	Secure Shell
SSL	Secure Sockets Layer
Subnet ID	Subnet Identifier
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Type Length Value
TTL	Time To Live
UDP	User Datagram Protocol
UNIX	Uniplexed Information and Computing System
VLAN	Virtual LAN
VOIP	Voice Over IP
WLAN	Wireless Local Area Network

Chapter 1

Introduction

”The Internet has become the global communication network, now is the time to sustain its growth and stability by integrating IPv6. IPv6 adds great value to IPv4" Dr. Vint Cerf

The first of January 1983 marked a tipping point in the History of the Internet. On that day, 400 computers connected through ARPAnet (Advanced Research Projects Agency Network) migrated from NCP (Network Control Program) to a new transport protocol that was better suited to serve a growing research environment, TCP/IP.

Developed by Vint Cerf, renowned today as one of the “fathers of the Internet”, this 32 bit protocol that allowed for a 4.3 billion address pool would be, according to Cerf, “*enough to conduct an experiment*”.

Twenty eight years later, this protocol, also known as the *Internet Protocol Version 4*, still remains as the dominant transport protocol in what is known today as the Internet, and neither has the “*experiment*” ended, nor is the world the same as it was back then.

With a growing, technologically knowledgeable population, massively spread services and applications, from refrigerators that send out warnings to your smart phone when you’re running out of milk, to cars that are virtual hot-spots on wheels, we have finally stretched IPv4 to the limit of its capacity.

On February 3rd 2011, in a public ceremony in Miami, IANA (the Internet Assigned Numbers Authority), the entity responsible for assigning addresses, assigned RIPE NCC – Network Coordinate Center (the Euro Zone RIR-Regional Internet Registries) the final /8 (185) block, thus marking the end of available addresses under the current IPv4 system.

Three decades ago, IPv4 and NCP co-existed throughout the year of 1982, leading up to the “flag-day”[1] in 1983. In all, a full year was necessary to properly transition a relatively small environment that encompassed only three major services (mail, FTP,

[1] “**Flag day”**, as used in system administration, is a change which requires a complete restart or conversion of a sizable body of software or data.

and telnet) to the new protocol. It comes as no surprise that for such a transition to occur today, with the wide spread of the Internet, would take decades.

To such an effect the IEFT (Internet Engineering Task Force) started the development of the Next Generation Protocol (IPng) relatively early, and it was this successful effort that led, in 1995, to the Protocol we know today as IPv6.

The approach of using different protocols in the same medium is known as “ships in the night”, as both protocols share the same hardware and/or software resources without ever knowing about each other or interacting on a protocol level.

This way, there is no need to change the existing IPv4 infra-structure, as both protocols can, and should, work alongside each other during a transitional period. The problem with this approach is that the first person that wants to “turn off” IPv4 will have to wait for the last person to “turn on” IPv6. As the new Internet Protocol is transparent to the end user, and its advantages are not immediately visible, everyone will most likely expect everyone else to step up first, both protocols will co-exist for years to come.

In a year when WLAN(Wireless Local Area Network)-capable devices are expected to hit the billion mark for the first time in history, and with previsions for that number to double by 2015, is the world ready for such a technological burst?

In this project, we discuss the differences between versions of the Internet Protocol, how it works, the transition, the evolution of technology and its users.

Objectives

Our aim was to deploy an IPv6 wireless network, implementing the necessary services and security measures, and test its compatibility with available Operating Systems.

Motivation

It was suggested to us that we contributed to the migration effort of our teaching institution by researching solutions and procure ways in which such effort could be mitigated, when it came to fruition. This project is the result of that labor, and we hope it can serve as a basis for an easier future implementation. It was developed in conjunction with another module that addresses IPLeiria’s IPv6 Network Migration.

Chapter 2

Why IPv6?

In this chapter we will address the main reasons behind the transition effort and limitations of the previous version. We all know IPv6 is supposed to have advantages over IPv6, but what exactly are they? Subjects such as address space, security issues, performance, and others deserve a certain degree of consideration.

2.1 IPv4 Limitations and Shortcomings

Saying a robust protocol like IPv4, which has been working perfectly for decades in the background of most people's everyday life, has any kind of shortcoming might be frowned upon by some, but the fact is that TCP/IP engineers and designers recognized the need for an upgrade as early as the late 1980's, when it became apparent that the continued growth of the Internet would soon enough strain the existing protocol.

This section introduces this factor, as well as other reasons for why an upgrade is due. These are:

- *Address Space Limitations*. The address crisis is the prime motivating factor for the upgrading efforts, as mentioned before.
- *Performance*. Although IP performs remarkably well, some of the design decisions made 30 years ago could stand some improvement.
- *Security*. With the expansion of the Internet, security has become one of the largest concerns for a network administrator.
- *Autoconfiguration*. Users would always prefer a "plug-and-play" feature, in opposition to the complex configuration of IPv4 nodes. Increased mobility of IP hosts also raises the issue of providing better support for configuring these hosts as they move across different networks [1].
- *Limitations of NAT (Network Address Translation)*. Developed as a stop-gap solution to the address exhaustion, there are some problems inherent to NAT.

Encryption of the TCP (Transmission Control Protocol) header, leads to complications implementing several NAT-sensitive applications and protocols.

2.1.1 Address Space Limitations

The IPv4 address space is a 32 bit field. As such, there are 4.294.967.296 unique values, considered in this context as a sequence of 256 "/8s", where each "/8" corresponds to 16.777.216 unique address values.

This might seem more than enough, but the actual number of possible addresses is less (in the case of public addresses, considerably less) because certain numbers are reserved or have some special significance.

Table 1 - IP address classes. [2]

Address Class	1st Octet Range	1st Octet Bits (bits in blue do not change)	Network(N) and Host(H)	Subnet Mask	Possible Networks and Hosts
A	1-127	00000000 - 01111111	N.H.H.H	255.0.0.0	128 nets (2^7) 16M hosts (2^{24} -2)
B	128-191	10000000 - 10111111	N.N.H.H	255.255.0.0	16,384 nets (2^{14}) 64K hosts (2^{16} -2)
C	192-223	11000000 - 110111111	N.N.N.H	255.255.255.0	2,097,150 nets (2^{21}) 254 hosts (2^8 -2)
D	224-239	11100000 - 11101111	Multicast	-	
E	240-255	11110000 - 11111111	Experimental	-	

As noted in RFC (Request For Comment) 5735, a number of address blocks are reserved for uses outside 'conventional' use in the public Internet as unicast identity tokens. In adding up these special purpose use address reservations there are the equivalent of 35.078 /8 address blocks in this category. This is composed of 16 /8 blocks reserved for use in multicast scenarios, 16 /8 blocks reserved for some unspecified future use, 1 /8 block (0.0.0.0/8) for local identification, a single /8 block reserved for loopback (127.0.0.0/8), and a /8 block reserved for private use (10.0.0.0/8). Smaller address blocks are also reserved for other special uses. The remaining 220.922 /8 address blocks are available for use in the public IPv4 Internet. [3]

There are occasionally arguments according to which the 16 /8s reserved in the experimental space could be used. Although this is likely to be possible for some IP stack implementations, for others it is not. At a minimum, some quick tests show that Windows 95 through Windows 2003 Server systems consider that block to be a configuration error and refuse to accept it.

Another debate occasionally resurfaces about reclaiming some of the early allocations to further extend the lifetime of IPv4, but following the allocation growth trend, after several years of litigation the result is likely to be just a few months of additional resource added to the pool - and possibly not even a whole month. [4]

Recently, and following the logic of appropriating the largest chunk of addresses left, Microsoft has purchased 666.624 addresses from Nortel Networks, which had been undergoing a process of bankruptcy since 2009. For this now valued commodity, MS (Microsoft) paid 7.5 million dollars (11.25 dollars per address).

2.1.2 IP Performance Issues

The Internet Protocol's initial purpose was to find optimal mechanics for moving data reliably, robustly and efficiently, linking dissimilar networks. It has, to a large extent, achieved this purpose. This does not mean it could not do without some improvements. Maximum transmission unit size, maximum packet size, header design, use of checksum, and much more has been addressed in this upgrade. [5]

2.1.3 IP Security Issues

In the old days of the ARPANet, when only reputable organizations, researchers and developers could connect to the Internet, security was not a major issue. Security was thought to be of less importance at the lower layers of the protocol stack, and security responsibility was passed upwards to the application. Therefore, IPv4 was designed with minimal security options.

In order to complement this, IPsec (Internet Protocol Security) has been used commonly to secure IPv4 traffic. What most are unaware of is that they are already reaping the benefits of the IPv6 effort. IPsec was, in fact, developed in conjunction with IPv6 and is, therefore, mandatory in all standards-compliant implementations of IPv6.

2.1.4 Autoconfiguration

In the early days, the "Internet" was static. There was no mobility, computers stood in the same room for years, and the Internet Service Provider industry was non-existant. Much has changed in the world, and what was once a fixed research network, is now highly mobile, and mainstream.

As such, there is now a growing population of users whose work styles require a greater mobility of network services. They may alternate from their laptop to their smartphone, work from home, and expect connectivity on-the-move.

This mobile workforce needs the ability to communicate with customers, partners, and fellow workers anywhere, anytime and have access to relevant business applications, tools to carryout business effectively. Enterprise mobility is about providing ubiquitous connectivity to the mobile user, independent of the devices and access technologies. [6]

This has led to some advances in the Version 4 protocol, introducing the Dynamic Host Configuration Protocol (DHCP), which allows systems to rely on servers to provide them with accurate IP network configuration.

But much more can be done, as the average user today expects true Plug-and-play capabilities, and the fact is IPv6 will be the way stateless configuration will be done in the near feature.

2.1.5 Limitations of NAT

NAT was originally developed as an interim solution to combat IPv4 address depletion by allowing globally registered IP addresses to be re-used or shared by several hosts. [7]

NAT conserves on the number of global IP addresses by providing translation from IP's in a non-routable private network, sitting behind a NAT device, to a single outside address. They also provide transparent packet forwarding between addressing realms. The packet sender and receiver remain unaware that NAT is taking place.

An enterprise can freely use these private addresses, and distribute them according to their own needs and preferences and in doing such, avoid obtaining registered public addresses.

Our need to conserve IPv4 addresses has prompted many to overlook the inherent limitations of NAT, recognized in RFC 1631 but deemed acceptable for a short-term solution. [7]

NAT regenerates TCP checksums, and here lies the root of the problem. If only the TCP payload is encrypted, as in SSH (Secure Shell) or SSL (Secure Sockets Layer), then the checksum can be recalculated. But, in the case of IPsec Transport Mode, since the TCP header is encrypted, the checksum field cannot be modified. Applications and protocols which use multiple TCP connections or UDP (User Datagram Protocol) streams to form session bundles, SIP (Session Initiation Protocol), or H.323 require the use of an ALG (Application Layer Gateway), a security component that augments the NAT employed in a network, in order to allow customized NAT traversal filters to be plugged into the gateway to support address and port translation. In order for these protocols to work through NAT or a firewall, either the application has to know about an address/port number combination that allows incoming packets, or the NAT has to monitor the control traffic and open up port mappings dynamically as required.

2.2 Developing Nations

Even though by the end of the 1990's, address exhaustion was already beginning to surface as a problem, no one could predict the technological boom that struck the Asia-Pacific region in the last decade. The number of Internet users is Asia is quickly surpassing the number of internet users in the rest of the world, accouting in 2009 for 43% of total users. [8]

Table 2- Internet Users and Population Statistics [8]

INTERNET USERS AND POPULATION STATISTICS				
Region	Population (2010)	Internet Users	User Growth (2000-2010)	Users % of World
Asia	3,834,792,852	**825,094,396**	621.8 %	42.0 %
Rest Of World	3,010,817,108	**1,141,420,420**	362.7 %	58.0 %

This region has been experiencing a digital explosion. On the back of massive increases in the sales of personal computers, China's online community – those who have regular access to the global internet – has been increasing at an astonishing rate. In 1996 it was estimated (exact figures were hard to come by) that there were 40,000 internet users in China. A year later it was 250,000. At this growth rate you would have expected the Chinese online community to have grown to more than three million by 2001. In fact reliable estimates suggest that by the end of 2001 the figure was more like 33.7 million, an unprecedented, breathtaking exponential growth rate. [9]

In fact, and according to Internet World Stats, an International website that features up to date world Internet Usage, Population Statistics and Internet Market Research Data, both China and India have experienced an increase in the order of 1500%, over the last ten years, in internet users.

In conclusion, the need for Next Generation IP protocol has, in our opinion, reached a tipping point. Measures to extend the life of IPv4 have long-since been implemented and exhausted, and the next logical step has to be full (all be it scaled) conversion to IPv6.

Chapter 3

IPv6 Header

In this chapter, we will delve into the structure of the IPv6 Header itself as well as the fields that comprise it.

3.1 The IPv6 Header

We begin by looking at the IPv6 header format, and how it differs from the IPv4 header. Designed with the future in mind, and even though 64-bit processors were rare at the time, the IPv6 header is optimized for 64-bit processing. **Word** is a term for the natural unit of data used by a particular processor. A word is basically a fixed sized group of bits that are handled as a unit by the processor. Because 64-bit CPUs (Central Unit Process) can read one 64-bit-wide memory word at a time, it is helpful that fields that are 64 bits (or a multiple of 64 bits) wide start at an even 64-bit boundary. Because every 64-bit boundary is also a 32-bit boundary, 32-bit CPUs aren’t affected negatively by 64-bit optimization. [10]

Figure 1 shows a comparison between headers of both versions.

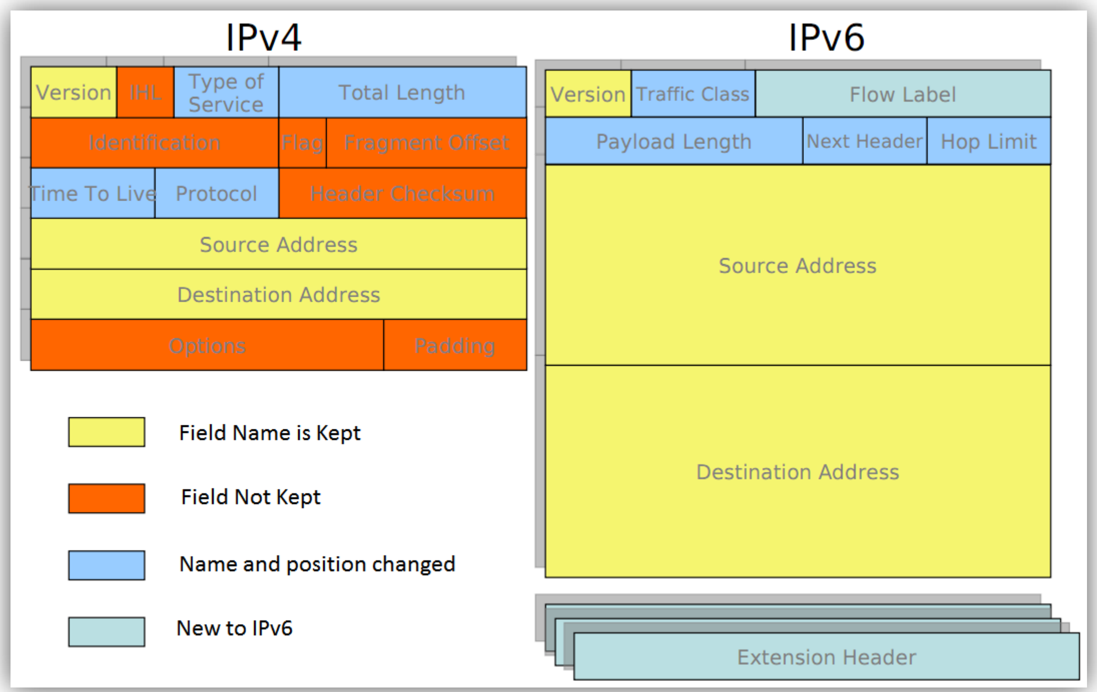


Figure 1- The IPv6 Header [11]

One feature that immediately comes to mind is the huge address space. This occurs because both Source and Destination addresses have been expanded from 32 bits to 128 bits. Despite this, the overall size of the header was not increased by a significant amount (20 bytes -> 40 bytes).

This is due to the simplification in the design. As shown in the diagram, fields marked in orange have disappeared in the new version. We go from 14 fields in the Version 4 to eight fields in Version 6.

Another major difference is that there is no Options field in IPv6. This field is used in version 4 to add information about various optional services. Because of this, the IP Header size fluctuates based on certain situations. IPv6 has a fixed header size, hence additional services information has been moved to the extended header section. The *Internet Header Length* (IHL) field that indicates the length of the IPv4 header is no longer needed because the IPv6 header is always 40 bytes long.

Header Checksum has also been eliminated. This is used to check for errors in header information. Since TTL (Time To Live) must be recalculated every time a packet flows through a router, a new checksum must also be recalculated for the new header content. By removing this redundant field (error checking is also performed in the above TCP layer), routers can be freed from these calculations. [1]

IP implements datagram fragmentation so that packets formed can pass through a link with a smaller maximum transmission unit (MTU) than the original datagram size. In IPv6, routers do not fragment, but drop the packets that are larger than the MTU. Fragmentation occurs only in the originating device, where a preceding ICMPv6 (Internet Control Message Protocol Version 6) message determines the minimum MTU along the path of the packet, thus eliminating the need for the *Identification*, *Flags*, and *Fragment Offset* fields.

3.2 The Fields

The fields that comprise the IPv6 header are described as follows:

- *Version* is a four-bit field that now contains 6 rather than 4.

- *Traffic Class* replaces the old *Type of Service*. This eight-bit value specifies a form of differentiated service, intended to enable scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop. RFC 1883 originally defined this field as only 4 bits long, and called it Priority field. The original semantics of the IPv4 Type of Service field have been superseded by the *diffserv* semantics per RFC 2474, which are also applied to IPv6;

- *Flow Label* field has a 20 bits length, and is a field newly established for IPv6. It is used to identify packets that belong to the same flow. By using this field, packet's sender or intermediate devices can specify a series of packets, such as Voice over IP (VOIP), as a flow, and request particular service for this flow. IPv4, like any packet-switched network, is designed to let each packet find its own optimal path to its destination. Since each packet finds its own way separately, packets sent from the same destination can be routed over entirely different paths. For more modern multimedia applications that are very sensitive to delays, such as audio or video streaming, having packets belonging to the same flow actually being routed together decreases that latency. Even in the world of IPv4, some communication devices are equipped with the ability to recognize traffic flow and assign particular priority to each flow. However, these devices not only need to check the IP layer information such as address of the sender and the destination, but also need to check the port number which is an information that belongs to a higher layer. Flow Label field attempts to put together all this necessary information and provide it at the IP layer;

- *Payload Length* is a 16 bit field that contains an integer value equal to the length of the packet payload in bytes. The *Total Length* is the length of the IPv4 packet including the header, but in IPv6, the *Payload Length* does not include the 40-byte IPv6 header, thereby saving the host or router receiving a packet from having to check whether the packet is large enough to hold the IP header in the first place—making for a small efficiency gain. Extensions are included as part of the payload for the purposes of calculating this field;

- *Hop Limit* replaces *Time To Live (TTL)*. Every time a node forwards a packet, it decrements this 8-bit field by one. When the field reaches zero, the packet is destroyed. This way, packets cannot circle the network forever when there are loops. Per RFC 791, the IPv4 TTL field should be decremented by the number of seconds that a packet is buffered in a router, but keeping track of how long packets are buffered is too difficult to implement, regardless of buffering time. The new name is a better description of what actually happens; [10]

- *Next Header* in IPv6 replaces the *Protocol* field in IPv4. In both cases, the field indicates the type of header that follows the IPv4 or IPv6 header. In most cases, the value of this field would be 6 for TCP or 17 for the *User Datagram Protocol* (UDP), but it may also indicate the existence of an IPv6 extension header;

- The *Source Address* is the 128 bit address of the node originating the packet;

- *Destination Address* is the 128 bit address of the intended recipient of the IPv6 packet. This address may be a unicast, multicast, or anycast address.

3.3 Extension Headers

After the main header in an IPv6 datagram, and before the encapsulated payload, the extension headers may be included if need be. This is an evolved form of the Options field in IPv4. The problem in IPv4 is that, being a field included in the main header, having different options means the header size will be different as well. This might not seem as much of a problem, but in fact this means special handling for these packets. Routers must optimize for performance, which means they are optimized for the most common form of packets, and not for these "special cases". The result is that IPv4 options tend to cause a router to put aside a packet, and deal with it later.

IPv6 Extension Headers drastically reduce the performance hit options cause in the routers, by not placing optional fields in the header itself, which is always fixed-sized, and moving them into the payload. This way, routers can forward optioned packets just as they forward non-optioned packets.

Chapter 4

IPv6 Addressing

The main directly visible difference from Version 4 to Version 6 has to be the addressing. While in ipv4 we had a familiar 32 bit addressing method, IPv6 addresses have 128 bits and are strange to the newly introduced observer. Extending the address space, along with optimization of the routing tables was the one of the main reasons behind the Next Generation Effort. The following chapter is as defined in RFC 4291, which obsoletes RFC 3513, RFC 2373, and RFC 1884 before that.

4.1 Addressing Model

IPv6 addresses of all types are assigned to interfaces, not nodes. An IPv6 unicast address refers to a single interface. A single interface can be assigned multiple IPv6 addresses of any type. Since each interface belongs to a single node, any of that node's interfaces' addresses may be used as an identifier for the node.

An IPv6 address consists of three parts, *Global Routing Prefix, Subnet ID*, and *Interface ID,* as defined in RFC 4291 and shown in Figure 2.

GLOBAL ROUTING PREFIX	SUBNET ID	INTERFACE ID
N bits	M bits	128 – M – N bits

Figure 2- Ipv6 Address Structure

The global routing prefix is used to identify a special address such as Multicast, or an address range. A subnet ID is used to identify a link within a site, and the interface ID is used to identify an interface on a link (and must be unique to that link).

4.2 Address Notation

IPv6 addresses have 128 bits, and are represented in 16-bit hexadecimal blocks separated by colons. There are three conventional forms for representing IPv6 addresses as text strings:

1. The preferred form is x:x:x:x:x:x:x:x, where each of the "x" are the hexadecimal values. For example:

   FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
   1080:0:0:0:8:800:200C:417A

   Note that even though it is not necessary to represent the leading zeros in an individual field, every field must at least contain one numeral.

2. It is common practice, in order to make writing addresses with large sequences of zeros easier, to compress them. The "::" notation can be used to compress leading, trailing or consecutive groups of zeros, but can only be used once per address.
   For example, these addresses can be represented as shown in Table 3.

Table 3 - Zero Compression

Type	No Compression	Zero Compression
A unicast address	1080:0:0:0:8:800:200C:417A	1080::8:800:200C:417A
A multicast address	FF01:0:0:0:0:0:0:101	FF01::101
Loopback address	0:0:0:0:0:0:0:1	::1
Unspecified addresses	0:0:0:0:0:0:0:0	::

3. In environments where IPv4 and IPv6 nodes co-exist, another convenient form of notation is to place the values of an IPv4 address into the four low-order byte pieces of address. x:x:x:x:x:x:d.d.d.d where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address and the 'd's are the standard IPv4 representation. For example, 0:0:0:0:0:0:192.168.0.2 can be written as ::192.168.0.2, or as ::C0A8:2.

   Generally speaking, an IPv6 address organizes itself as shown in Figure 3.

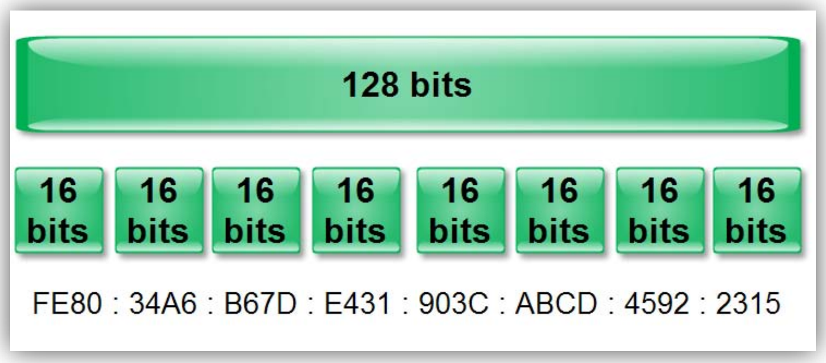


Figure 3 - Address Notation in IPv6

4.2.1 Problems with IPv6 Text Representation

Several problems have been encountered with this flexible model of text representation of IPv6 addresses. RFC 4291 does not, for example, reference any preference of uppercase or lowercase of the hexadecimal characters. This problem was never encountered in IPv4, due to its decimal notation. The zero compression can also cause the same address to be represented in different ways. There are also different ways to combine IPv6 addresses with port numbers, which can lead to various problems due to the colon already being used to separate 16-bit fields. The following is a list of processes or procedures in which such problems could happen.

- Searching. A search of an IPv6 address if conducted through a UNIX (Uniplexed Information and Computing System) system is usually case sensitive and extended options that allow for regular expression use will come in handy. However, there are many applications in the Internet today that do not provide this capability. When searching for an IPv6 address in such systems, the system engineer will have to try each and every possibility to search for an address. This has critical impacts, especially when trying to deploy IPv6 over an enterprise network [12];
- Parsing and Modifying. With all the possible methods of text representation, each application must include a module, an object, a link, etc. to a function that will parse IPv6 addresses in such a manner such that no matter how it is represented, they will mean the same address. Many system engineers who integrate complex computer systems for corporate customers will have difficulties finding that their favorite tool will not have this function, or will encounter difficulties such as having to rewrite their macros or scripts for their customers [12];
- When an operator sets an IPv6 address of a system as 2001:db8:0:0:1:0:0:1, the system may take the address and show the configuration result as 2001:DB8::1:0:0:1. Someone familiar with IPv6 address representation will know that the right address is set, but not everyone may understand this [12].

4.2.2 A Recommendation for a Canonical Text Representation

This recommendation is as specified in RFC 5952, which addresses IPv6 text representation, and as such, updates and is in accordance with RFC 4291. It is a recommendation that should be followed by any system, and it is advised that humans also follow this representation when spelling.

- *Handling Zeros*. Leading zeros MUST be suppressed. For example, 2001:0db8::0001 is not acceptable and must be represented as 2001:db8::1. A single 16-bit 0000 field MUST be represented as 0. The symbol “::” must be used to its maximum capability. However, the double colon MUST NOT be used to shorten just one 16-bit field. For example, the representation 2001:db8:0:1:1:1:1:1 is correct, but 2001:db8::1:1:1:1:1 is not correct. Also, the longest run of consecutive 16-bit zero fields MUST be shortened;
- *Lowercase*. The characters "a", "b", "c", "d", "e", and "f" in an IPv6 address MUST be represented in lowercase;
- *Port Numbers*. The [ ] style SHOULD be used. For example, [2001:db8::1]:80.

4.3 Prefix Notation

IPv6 prefix representation is similar to that of IPv4, written in CIDR (Classless-Inter-Domain-Routing) notation. A prefix is the high-order bits of an IPv6 address used to identify the subnet or a specific type of address. An IPv6 prefix is represented as follows:

Ipv6-address/prefix-length (example: 2001:0DB8:0:CD30::/60)

The prefix length is a decimal value specifying how many of the leftmost bits of the address comprises the prefix. The prefix is used to identify the subnet that an interface belongs to and is used by routers for forwarding.

4.4 Address Types

IPv6 addresses are comprised of three types: unicast, anycast, and multicast. Broadcast addresses have been deprecated. Broadcasts were intended to be used to carry information to more than one node or for making requests when the requesting node doesn't know where that information may come from. However, broadcasts have become a cumbersome type of unnecessary traffic on a specific link. All nodes are required to process them, and more than likely ignore them, for not being relevant.

The end of the broadcast signified the end of ARP (Address Resolution Protocol), which used broadcast to send packages to every host in a given collision domain in order to retrieve the MAC (Media Access Control) address of the packages destination. This protocol has been replaced with Neighbor Discovery protocol which uses specific ICMPv6 messages.

The IPv6 solution consists in using an "all nodes" multicast address to replace those broadcasts that are absolutely necessary, while resorting to more limited multicast addresses for other situations. This way, interested nodes can subscribe to a multicast address, while other can choose to ignore them. Broadcasts never adequately solved the problem of propagating information across the Internet [1].

The type of an IPv6 address is identified by the high-order bits of the address, as follows:

Table 4 - IPv6 Address Types

Address Type	Binary Prefix	IPv6 notation
Unspecified	00…0 (128 bits)	::/128
Loopback	00…1 (128 bits)	::1/128
Multicast	11111111	FF00::/8
Link-Local Unicast	1111111010	FE80::/10
Global Unicast	(everything else)	

4.4.1 Unicast

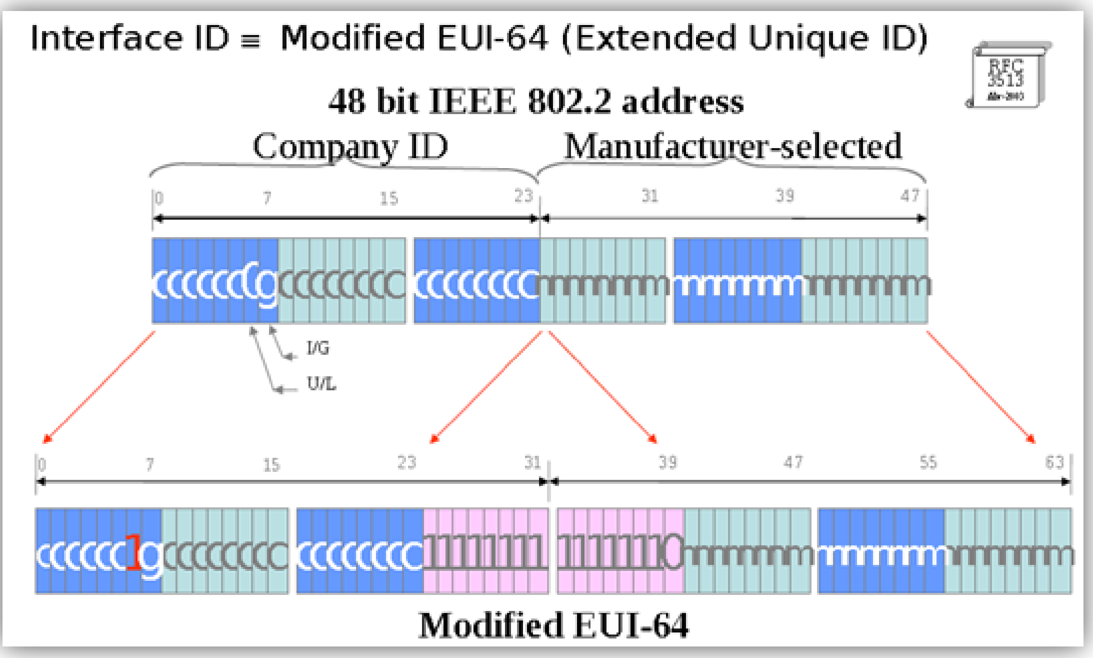
Unicast addressing is used for the vast majority of internet traffic under IPv4, and the same will be applicable under Version 6. 1/8th of the total addressing space of IPv6 is assigned to unicast addresses.

Unicast addresses identify a single interface. Since each interface belongs to a node, which can hold multiple interfaces, any of that node's address can be used to identify it. There are several types of Unicast addresses, as follow.

- Interface Identifiers

Interface Identifiers are used to identify interfaces on a link, and are required to be unique within a subnet prefix. An interface identifier is very much like the 48-bit MAC address. These addresses are specific to each network interface card, and are burned into them by the manufacturer. For all unicast addresses, except those that start with the binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 (Extended Unique Identifier) format.

Modified EUI-64 format interface identifiers are formed by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology) when forming the interface identifier from IEEE EUI-64 identifiers. In the resulting Modified EUI-64 format(Figure 4), the "u" bit is set to one (1) to indicate universal scope, and it is set to zero (0) to indicate local scope [13].



Interface ID ≡ Modified EUI-64 (Extended Unique ID)
RFC 3513
48 bit IEEE 802.2 address
Company ID
Manufacturer-selected
0
7
15
23
31
39
47
cccccc0gcccccccc cccccccc mmmmmmmm mmmmmmmm mmmmmmmm
I/G
U/L
0
7
15
23
31
39
47
55
63
cccccc1gcccccccc cccccccc11111111 11111110mmmmmmmm mmmmmmmm mmmmmmmm
Modified EUI-64

Figure 4 - EUI64 [11]

- Unspecified Address

The unspecified address is the "all-zeros" address. It is used, for example, when an initializing host that has not yet been assigned an address and sends out a request for configuration, placing 0:0:0:0:0:0:0:0 in the source field of the IPv6 Header. It must never be assigned to a node. It is also usually represented as "::" .

- Loopback Address

In IPv4, the loopback address is defined as 127.0.0.1; it refers to the network interface itself. Any packets which contain the loopback address as a destination must travel down the protocol stack, to the interface. The interface itself processes these packets as if they originated from outside the node, and pass them back up the stack. In IPv6, the loopback address is all-zeros, except for the lowest order bit. In other words, it is represented as 0:0:0:0:0:0:0:1 or ::1.

- Link Local Address

Link local addresses are intended to be used to number hosts on a single network link. They are used for automatic address configuration, neighbor discovery, or when a router is not present. Routers can safely ignore packets with a link-local source or destination addresses, for these packets are never to be forwarded.

LINK LOCAL PREFIX 10 bits	0000…0000 54 bits	INTERFACE ID 64 bits

Figure 5 - Link Local Unicast Address

As can be seen from Figure 5, The first ten bits identify the address as link-local (FE80::/10, or 1111111010), the middle 54 bits are set to zero, and the interface ID uses the same EUI-64 implementation as described earlier [13].

- Global Unicast Address

The general format of Global Unicast Addresses is demonstrated in Figure 6.

GLOBAL ROUTING PREFIX 48 bits	SUBNET ID 16 bits	INTERFACE ID 64 bits

Figure 6 - Global Unicast Address

The part of the IPv6 address space set aside for unicast addresses is structured into an address format that uses the first 48 bits for the *routing prefix* (like a network ID), the next 16 bits for a *subnet ID*, and the final 64 bits for an *interface ID*. Due to this structure, most end sites (regular companies and organizations, as opposed to Internet service providers) will be assigned IPv6 networks with a 48-bit prefix. These are commonly referred to as 48s, or /48s. [14]

- IPv6 Addresses with Embedded IPv4 Addresses (IPv4 compatible and IPv4-mapped)

The transition to IPv6 will always be a gradual one. In order to assist in this transition, and as specified in RFC 4291 and RFC 4038, two types of IPv6 unicast addresses were devised. They are the IPv4-Compatible IPv6 address, and the IPv4-Mapped IPv6 address. The former is now deprecated and was meant to be used by nodes that need to tunnel IPv6 packets through, as the latter is used by IPv6 nodes to address nodes that support only IPv4. The format of IPv4-Mapped IPv6 address is exemplified in Figure 7.

0000…0000 80 bits	FFFF 16 bits	IPv4 Address 32 bits

Figure 7 - IPv4-Mapped IPv6 Address

4.4.2 Multicast

As mentioned earlier, broadcast as a specific addressing type has been eliminated in IPv6. Broadcasts were easy, as there was no decision to be made. If it was a broadcast, the node listened to it.

Multicasts are a bit more complicated. The node subscribes to a multicast address, and if it senses that the destination of the packet is that of the multicast address which it has subscribed to, it will then forward it. So, a multicast address identifies a group of interfaces to which a packet will be delivered to.

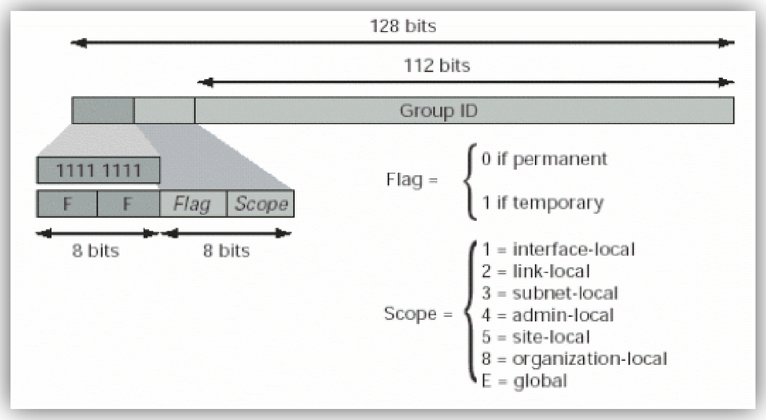


Figure 8 - Multicast Address

Multicast addresses adhere to a rigid format, and must always be used only as destination addresses. No packet should ever be originated with a multicast address as its source address. Multicast addresses comprise a full 1/256th of the IPv6 address space. As such, the first eight bits are always set to 1 (Figure 8), to indicate a multicast address.

Some well-known Multicast Addresses include:

Table 5 - Well-known Multicast Addresses

Address Type	Address
All-node	FF02:0:0:0:0:0:0:1
All-routers	FF02:0:0:0:0:0:0:2
RIP	FF02:0:0:0:0:0:0:9
EIGRP	FF02:0:0:0:0:0:0:A
Solicited Node	FF02:0:0:0:0:1:FFXX:XXXX

In addition to the regular multicast addresses, each unicast address has a special multicast address called its *solicited-node address*. This address is created through a special mapping from the device's unicast address. Solicited-node addresses are used by the IPv6 Neighbor Discovery (ND) protocol to provide more efficient address resolution than the ARP technique.

All solicited-node addresses have their *T* flag set to zero and a scope ID of 2, so they start with "FF02"[14], as can be seen in Figure 9.

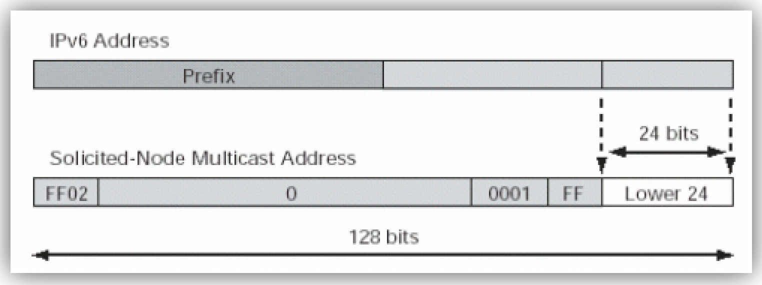


Figure 9 - The Solicited Note Address

4.4.3 Anycast

Anycast addresses are a new type of address introduced in IPv6, and they are considered to be a cross between unicast and multicast. Where unicast says “send to this one address” and multicast says “send to every member of this group”, anycast says “send to any one member of this group”. [14]

An anycast address is an address that is assigned to more than one interface (typically belonging to different nodes), and functions very much like multicast, with the particularity that a packet sent to an anycast address is routed to the nearest interface having that address, according to the routing protocols’ measure of distance.

This is particularly useful when providing certain types of services. DNS (Domain Name) servers and NTP (Network Time Protocol) servers fall into this category, as the closest name server should function just as well as the farthest one. Likewise, a closest NTP server is always preferable for accuracy purposes.

Anycast addresses are allocated out of normal unicast space. As such, they cannot be differentiated from unicast addresses in their format.

Chapter 5

IPv6 Authentication and Security

Nowadays many security threats exist throughout the internet and these threats are increasing daily.

IPv6 brings some new security features when compared to IPv4. Let's therefore see the mechanisms involved in this new protocol and also the new security issues, which may arise.

5.1 Types of Threats

The IT security threats can be classified in four principle archetypical attacks as follows:

- Disruption of service
- Modification
- Fabrication
- Eavesdropping

A *disruption* or *denial of service* is usually easy to recognize, but it can be hard to determine the real cause of the problem. In a brute-force attack (i.e., physical destruction through fire or violent acts), significant resources must be spent on repairing or replacement of damaged equipment. A subtle variation of service disruption is the (hidden) degradation of quality of service, such as the introduction of artificial communication delays, which may disturb the proper execution of a business process but not be perceived as an attack [5].

The *fabrication, modification or deletion of information* is much harder to detect or defend against than service disruption, unless specific protection mechanisms are in place. A broad spectrum of attack possibilities exists, ranging from the modification of individual data elements (such as the sending time of an email message) to the insertion of falsified payment orders through masquerading, the distribution of a virus, or the complete deletion of database or log files [5].

Electronic eavesdropping, or the picking up and evaluation of information, may be carried out in a variety of ways, from classic wiretapping to the usage of Trojan horses on systems under the attacker's control or the gathering of electronic radiation emanating from devices such as screens, printers, telephones, encryption devices, or video cards. Such passive attacks are usually impossible to detect [5].

5.2 IPsec

The security framework for the IP protocol layer has been formally defined and standardized by the IETF IP Security Protocol Working Group (IPSEC) in RFC 2401.

Figure 10 exhibits the IPsec components.

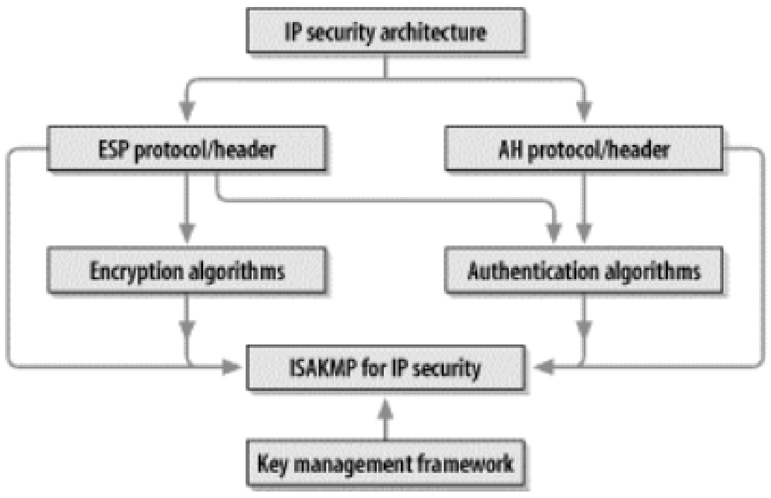


Figure 10 - Relationship between IPSEC components (RFC 2411)

As we can see, IPsec consists of six elements as follows:

- A general description of security requirements and mechanisms on the network layer ;
- A specific security element for encryption (Encrypted Security Payload [ESP]; RFC 2406);
- A specific security element for authentication (Authentication Header [AH]; RFC 2402) ;
- Definitions for using concrete cryptographic algorithms for encryption and authentication;
- The definition of security policies and Security Associations between communicating partners;
- IPSEC key management, which is based on key management on a more general framework.

These mechanisms can either be used in IPv4 or IPv6. But they are an integrated mandatory part of the basic IPv6 protocol suite.

5.3 802.1X Architecture

The protocol 802.1x is used for authentication purposes in IEEE 802 networks and can also be used in an IPv6 environment.

Figure 11exhibits the components involved in 802.1x authentication:

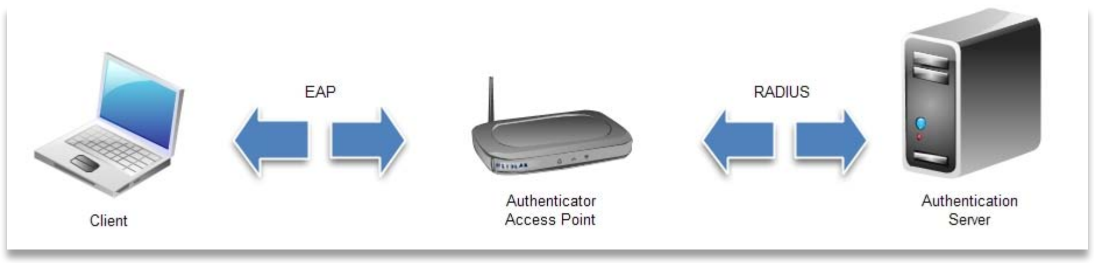


Figure 11 – 802.1X components

In our tests, we used this concept to test the authentication mechanisms in our network.

5.3.1 Authentication process

The client designated as *supplicant* is the one who will access the network.

The authenticator is the one who controls the network access. The client and the authenticator are designated as Port Authentication Entity (PAE)

The authenticator (Access Point or Switch) acts like a *relay* between the client and the authentication server. The client's requests to the authenticator are forwarded to the authentication server. According the server's answer, the authenticator will enable the access to the client or not.

The communication between the client and the authenticator is done through the EAPOL protocol (EAP over LAN). The communication between the authenticator and the authentication server uses the RADIUS protocol (Remote Authentication Dial-In User Service).

5.3.2 RADIUS

RADIUS is an authentication protocol. It consists in creating an encrypted tunnel for sending information using the encryption TLS (Transport Layer Security) protocol.

There are four types of RADIUS packets:

- Access Request : it starts an authentication process with RADIUS
- Access-Accept : it indicates that the client's credentials are valid
- Access-Reject: it indicates that the client's credentials are not valid
- Access-Challenge: it is used to send a *challenge* to the RADIUS client in order to inform about the credentials.

The necessary messages sequence (Figure 12) for a client authentication only starts after establishing an association between the client and the access point. As soon as the association is created, the authentication process can start:

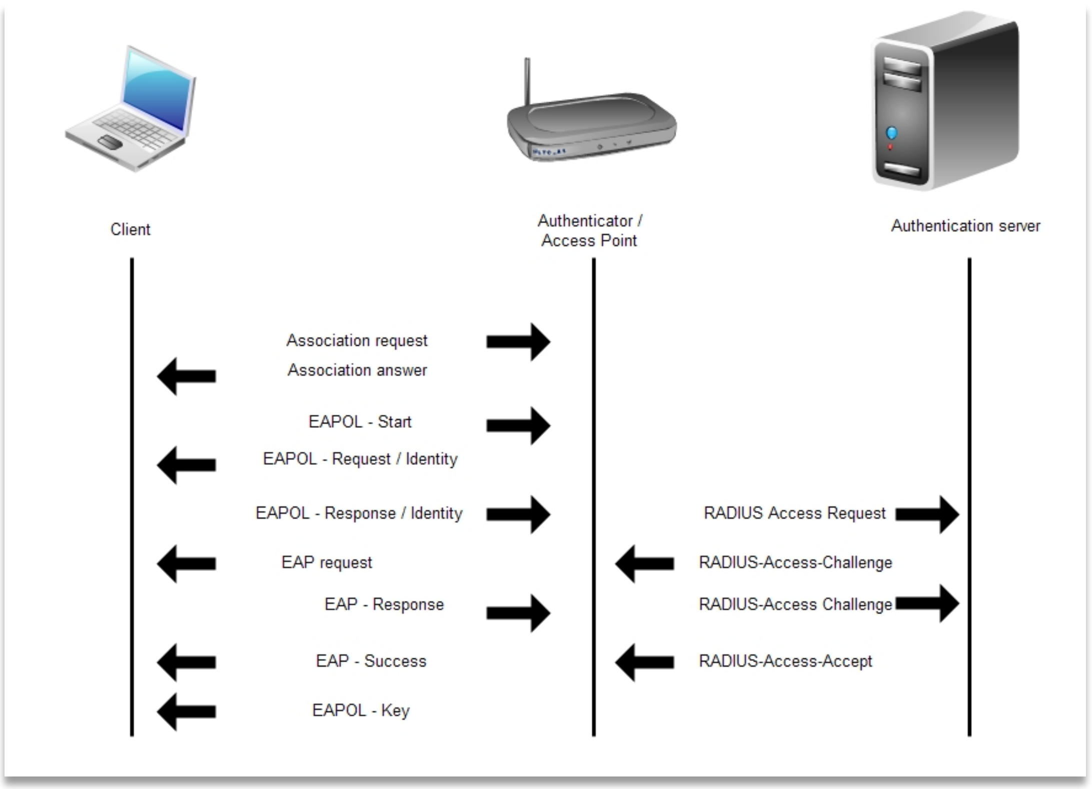


Figure 12 – Authentication message sequence

5.4 Open issues

Despite the IPv6 integrated security features, some security issues may arise.

Scanning for valid host addresses and services is considerably more difficult in IPv6 networks than it is in IPv4 networks, and to effectively scan a whole IPv6 segment may take millions of years. However, it doesn't mean that having a larger address makes IPv6 less vulnerable to flooding issues. Not even the lack of broadcast addresses makes IPv6 more secure. New features like multicast addresses continue to be source of problems.

IPv6 mobility uses the real address and the mobile address. The real address is a typical IPv6 address contained in an "extension header". On the other hand, the mobile address is a temporary address contained in the IP header. Because of the characteristics of these networks, the temporary mobile address is more vulnerable to spoofing attacks on the home agent.

There is hardly any doubt that IPv6 will bring considerable improvements compared to the old IPv4 protocol stack. It provides several features that improve not only the overall functionality, but also some specific security functions. But it would be a mistake if it was considered to be an ultimate solution. Although IPv6 offers better security features like larger address space and the use of encrypted communication (IPsec), the new protocol also raises significant new security challenges [15].

Chapter 6

Auto-Configuration

The ability to auto-configure (the means by which a host client configures itself with the necessary options required in order to establish a connection to a network) is, in a mobile environment, indispensible. A network administrator can't possibly expect every user to manually configure his or her equipment with IP and DNS settings, even in a SOHO (Small Office Home Office) network, let alone on the internet.

Every TCP/IP network requires that all hosts have unique addresses to facilitate communication. When a network is manually configured with a distinct IP address for each host, the hosts permanently know "who they are". When hosts are made DHCP clients, they no longer have a permanent identity; they rely on a DHCP server to tell them "who they are" [14].

Now when it comes to DHCP for IPv6 (as stated in RFC 3315), this method is known as stateful auto-configuration and is centrally managed on DHCP servers. The DHCP clients use stateful DHCP to obtain an IP address and other useful configuration information from the DHCP servers.

What is new in IPv6 is the Stateless Address Auto-Configuration, also known as SLAAC.

6.1 Stateless Address Auto-Configuration

IPv6 Stateless Address Auto-configuration is defined in RFC 4862. Simply put, IPv6 hosts configure themselves automatically when connected to an IPv6 network using Internet Control Message Protocol version 6 (ICMPv6) Neighbour Discovery messages.

The Neighbor Discovery Protocol (NDP) is a protocol in the Internet Protocol Suite used with IPv6. It operates at the Network Layer of the Internet model (RFC 1122) and is responsible for address auto-configuration of nodes, discovery of other nodes on the link, determining the Link Layer addresses of other nodes, duplicate address detection, finding available routers and Domain Name System (DNS) servers, address prefix discovery, and maintaining reachability information about the paths to other active neighbor nodes [16].

NDP is defined in RFC 2461. It uses ICMPv6 to exchange the messages necessary for its functions; specifically, five new ICMPv6 messages are pointed out in RFC 2461:

- **Router Advertisement** (RA) messages are originated by routers to advertise their presence and link-specific parameters such as link prefixes, link MTU, and hop limits. These messages are sent periodically, and also in response to Router Solicitation messages.
- **Router Solicitation** (RS) messages are originated by hosts to request that a router send an RA.
- **Neighbor Solicitation** (NS) messages are originated by nodes to request another node's link layer address and also for functions such as duplicate address detection and neighbor unreachability detection.
- **Neighbor Advertisement** (NA) messages are sent in response to NS messages. If a node changes its link-layer address, it can send an unsolicited NA to advertise the new address.
- **Redirect** messages are used the same way that redirects are used in ICMP for IPv4; they have merely been moved from being a part of the base ICMPv6 protocol to being a part of NDP.

When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; if configured suitably, routers respond to such a request with a router advertisement packet that contains network-layer configuration parameters [17].

The following steps are typically performed during auto-configuration.

1. The node begins the process by generating a link-local address for the interface, derived from the MAC address of the Interface.
2. The node sends a neighbor solicitation message containing the previously generated link-local address, in order to determine if a duplicate address exists on the link.
3. After determining that the address is unique, it is assigned to the interface.
4. At this point, the node has IP connectivity with neighbor nodes.

The next phase of auto-configuration involves obtaining a router advertisement or determining that no routers are present. If routers are present, the routers send router advertisements that specify what type of auto-configuration a host should perform. Routers send router advertisements periodically. However, the delay between successive advertisements is generally longer than a host that performs auto-configuration can wait. To quickly obtain an advertisement, a host sends one or more router solicitations to the all-routers multicast group. Because routers periodically generate router advertisements, hosts continually receive new advertisements. IPv6-enabled hosts process the information that is contained in each advertisement. Hosts add to the information. They also refresh the information that is received in previous advertisements [18].

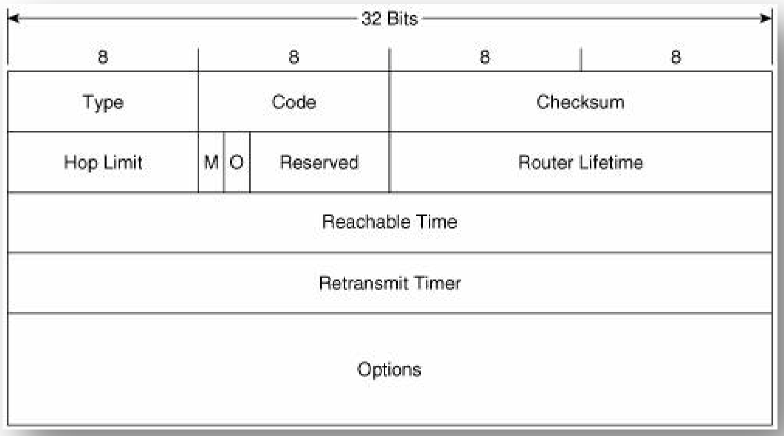


Figure 13 - Router Advertisement

Of all the fields in Figure 13, the ‘M’ and ‘O’ flags need careful appreciation. ‘M’ is the *Managed Address Configuration* flag. If this bit is set, the originating router is telling hosts on the link to use stateful address auto-configuration via DHCPv6. If the flag is cleared, hosts on the link should use stateless address auto-configuration. Address auto-configuration is described later in this chapter.

O is the *Other Stateful Configuration* flag. When set, the originating router is telling hosts on the link to use DHCPv6 for the acquisition of other link information.

The ‘M’ and ‘O’ flags can be used together. For example, by clearing the ‘M’ flag but setting the ‘O’ flag, the router is telling hosts to use stateless address auto-configuration but then consult a DHCPv6 server for other configuration parameters. The combination of Stateless and Stateful configuration will be the subject of testing later in this project.

The auto-configuration capability of IPv6 has been designed to ensure that manual configuration by hosts before connecting to the network is not required. The interface ID is obtained by converting the MAC address resulting in a EUI64 Interface ID, using a mechanism called MAC-to-EUI64.

As a result, a prefix needs to be added to the interface ID. The prefix can either be announced through a Router Advertisement, or by using a default link local prefix FE80.

If the host only needs to communicate with devices on the same link, auto-configuring its link-local address is sufficient. But if it needs to communicate with devices outside the link, then it needs an address with a wider scope normally a global IPv6 address, which can be obtained by a stateless or stateful address auto-configuration process (using DHCPv6).

6.2 Stateful Address Auto-Configuration

As mentioned earlier in this chapter, a stateful address configuration is obtained by using DHCPv6. The purpose of the new version is much the same as that of version 4, and as such we will only address the differences in the new protocol.

DHCP is based on an earlier protocol called BOOTP. This packet layout is wasteful in a lot of cases. A lot of the options turn out to be not useful, or not as useful as they can be, but it is hard to change a protocol with such a large installed base. There are a lot of "tweaks" that implementations need in order to be compatible with the buggy clients. DHCPv6 leaves all this behind.

Two features of IPv6 greatly improve DHCPv6: IPv6 hosts have "linklocal addresses". Every network interface has a unique address that can be used to send and receive exclusively on that link. IPv6 hosts can use this to send requests for "real" addresses. IPv4 hosts have to use system specific hacks to work before they have an address. Also, all IPv6 systems support multicasting. All DHCPv6 servers register that they want to receive DHCPv6 multicast packets. This means the network knows where to send them. In IPv4, clients broadcast their requests, and networks do not know how far to send them [17].

Chapter 7

Auto-configuration Tests

One of the main necessities in deploying a wireless network is the urgency in creating means by which clients can auto-configure themselves, which is, as mentioned in chapter 2, one of IPv6’s advantages. The user is not always tech-knowledgeable, and cannot be expected to configure his/her machine with the necessary fields.

As mentioned earlier, there are two modes of auto-configuration: stateless and stateful. In this chapter, the available implementations of each were tested.

For these tests, we initially used Ubuntu Server v10.10. Ubuntu Server is a secure and powerful Debian-based open source operating system designed to be used as a platform for running essential network (and other) services. It was chosen in this project as a first test-bed due to its extensive documentation and availability of the latest versions of certain *daemons*, which will be described throughout this chapter.

At this first stage, we chose to run the server as a Virtual Machine, as this allowed for safer trial and error testing.

The scenario in Figure 14 served as a testbed for the following configuration tests.

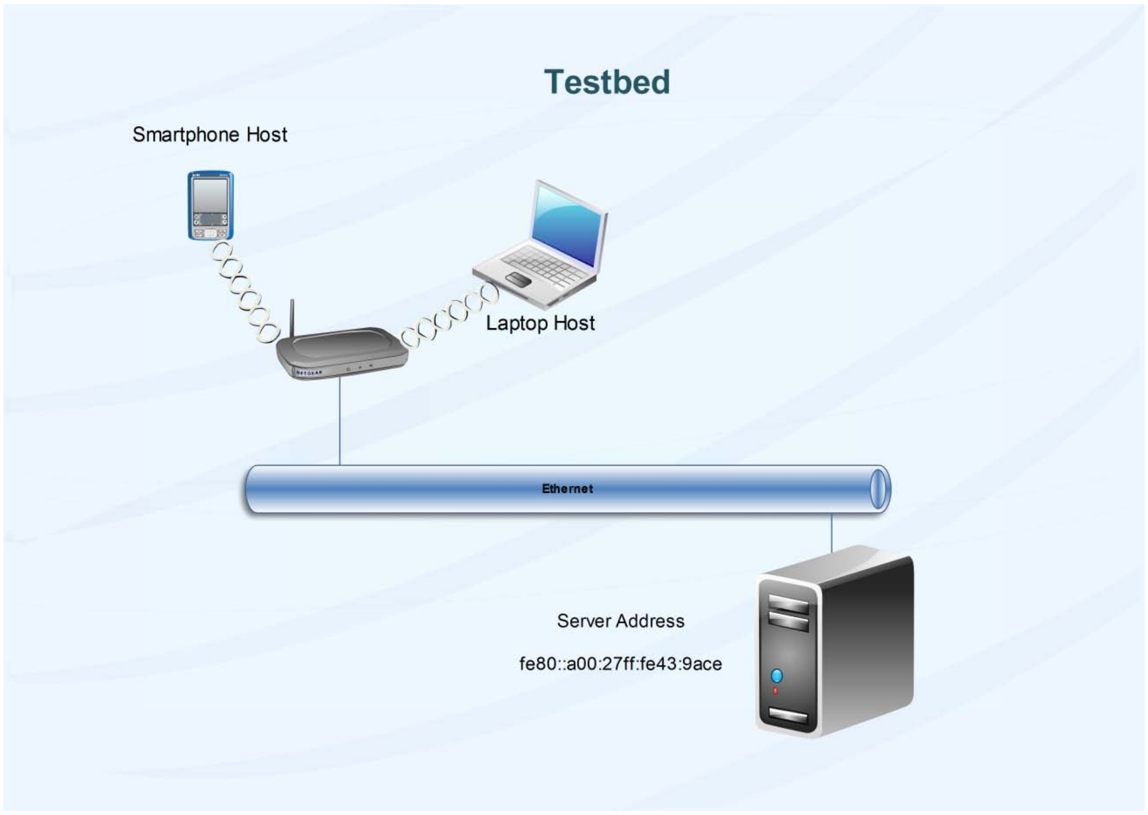


Figure 14 – Auto-configuration testbed

The server responsible for running all the different daemons has a link-local address of ‘fe80::a00:27ff:fe43:9ace’. This will be relevant when inspecting packet traffic. The choice of using an Access Point only to forward traffic at the network layer was due to limitations of available equipment regarding its IPv6 readiness. As no compatibility with IPv6 existed, this way the access point is only responsible for switching traffic at the data-link level.

All through this chapter, the following OS’s were tested as hosts:

Table 6 – Tested Host Operating Systems

Operating System	Ubuntu Desktop	Fedora	Android	Mac OS X	Microsoft Windows	Microsoft Windows Mobile
Version(s)	10.04 LTS 11.04	15	2.2 2.2.1 2.3	Leopard	XP Vista Seven	6.1

7.1 Stateful Configuration of Hosts

The basic DHCPv6 client-server concept is similar to using DHCP for IPv4. If a client wishes to receive configuration parameters, it will send out a request on the attached local network to detect available DHCPv6 servers. This is done through the "Solicit" and "Advertise" messages. Well known DHCPv6 Multicast addresses are used for this process. Next, the DHCPv6 client will "Request" parameters from an available server which will respond with the requested information with a "Reply" message. Figure 15 demonstrates the sequence of events [18]:

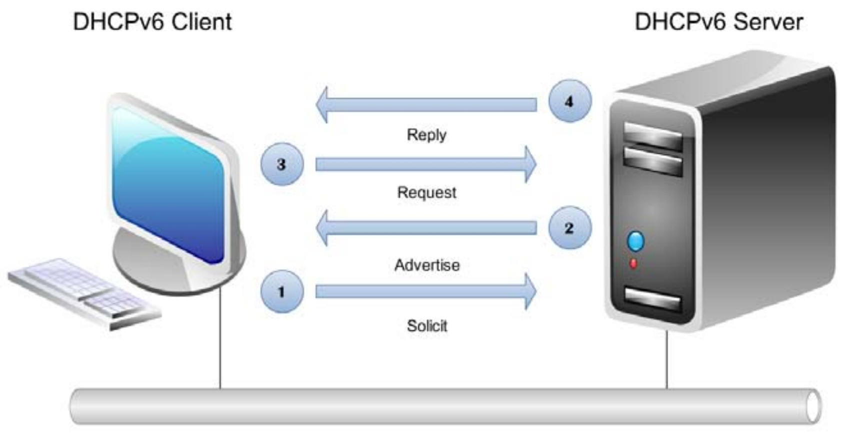


Figure 15 - DHCPv6 Message Exchange

Of the several daemons available for DHCPv6, we chose WIDE DHCPv6 Server and Dibbler, as they are both very well documented.

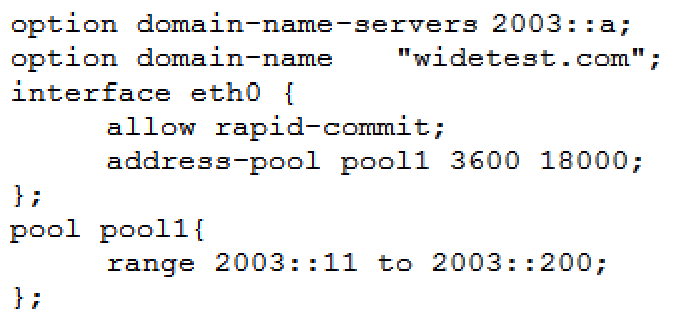
7.1.1 WIDE DHCPv6

WIDE-DHCPv6 is an open-source implementation of DHCPv6, originally developed by WIDE (Widely Integrated Distributed Environment), a Japanese consortium of companies and public institutes whose main objective was Research and Development into IPv6. It is now maintained separately, and its purpose was to release, develop and maintain an up-to-date IPv6 implementation for BSD and Linux. It provides some level of stateless configurations, as it can advertise DNS, NTP, and SIP servers [19].

Installing the server daemon (Debian):

```
$apt-get install wide-dhcpv6-server
```

The daemon is configured by editing the file "/etc/wide-dhcpv6/dhcp6s.conf". The following modifications were made as shown in Figure 16.



```
option domain-name-servers 2003::a;
option domain-name    "widetest.com";
interface eth0 {
      allow rapid-commit;
      address-pool pool1 3600 18000;
};
pool pool1{
      range 2003::11 to 2003::200;
};
```

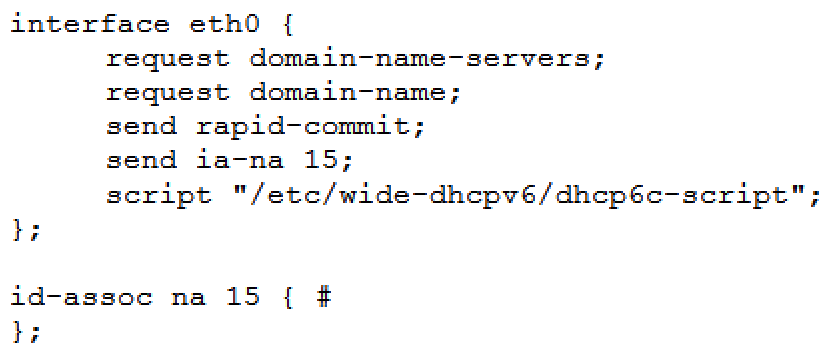
Figure 16 - dhcp6s.conf configuration file

This way, DNS options are to be advertised, as well as a pool of 2003::11 to 2003::200 for addressing. What we discovered was that even though the host's Operating Systems were configured to obtain IPv6 addressing automatically, they would not send out DHCP Solicit messages, unless an actual client was installed, in this case *wide-dhcpv6-client*. Unfortunately, it's only available for Linux distributions.

Installing the client daemon (Debian):

```
$apt-get install wide-dhcpv6-client
```

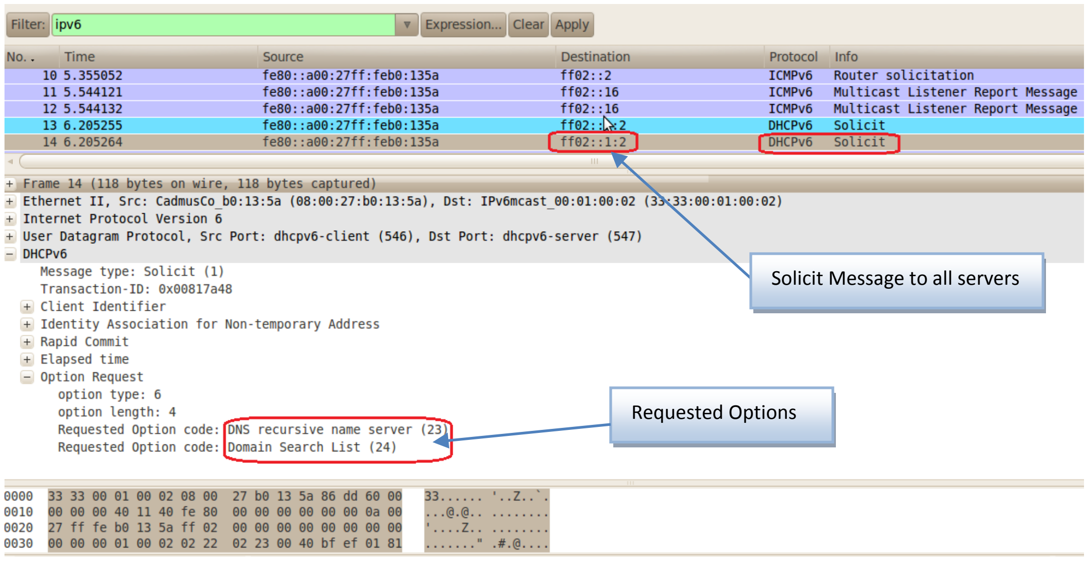
Figure 17 shows the host-side configurations, as made in the Linux hosts tested:



```
interface eth0 {
      request domain-name-servers;
      request domain-name;
      send rapid-commit;
      send ia-na 15;
      script "/etc/wide-dhcpv6/dhcp6c-script";
};

id-assoc na 15 { #
};
```

Figure 17 - dhcp6c.conf configuration file

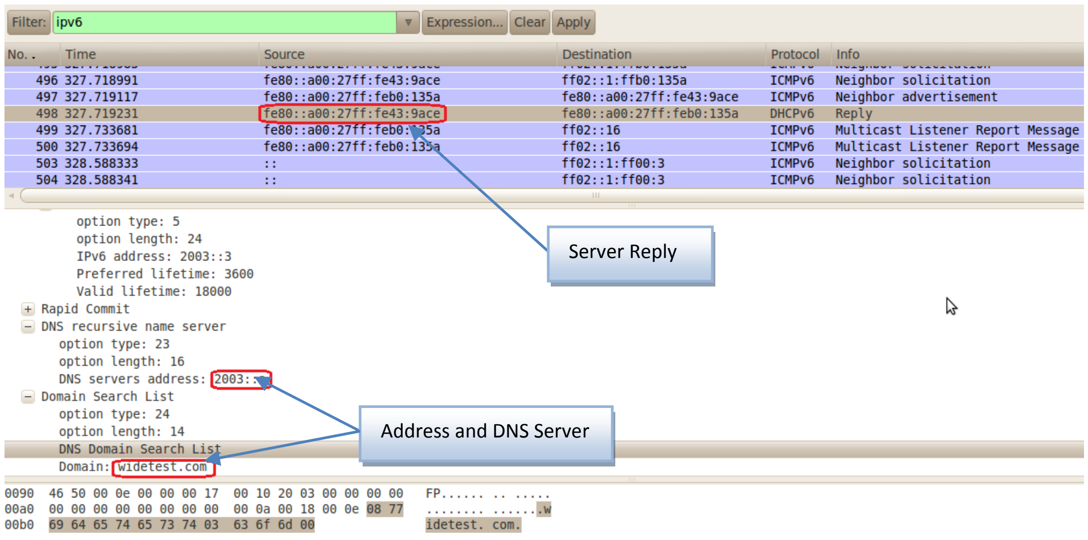
This will force the client to request a permanent address (through the "*send ia-na*" command), and trigger a DHCP solicit message to be sent to a well-known multicast address (representing all DHCP servers and relay agents on the local network site), as it is made clear by Figure 18.



```
No. .  Time        Source                       Destination   Protocol  Info
    10 5.355052    fe80::a00:27ff:feb0:135a     ff02::2       ICMPv6    Router solicitation
    11 5.544121    fe80::a00:27ff:feb0:135a     ff02::16      ICMPv6    Multicast Listener Report Message
    12 5.544132    fe80::a00:27ff:feb0:135a     ff02::16      ICMPv6    Multicast Listener Report Message
    13 6.205255    fe80::a00:27ff:feb0:135a     ff02::1:2     DHCPv6    Solicit
    14 6.205264    fe80::a00:27ff:feb0:135a     ff02::1:2     DHCPv6    Solicit

+ Frame 14 (118 bytes on wire, 118 bytes captured)
+ Ethernet II, Src: CadmusCo_b0:13:5a (08:00:27:b0:13:5a), Dst: IPv6mcast_00:01:00:02 (33:33:00:01:00:02)
+ Internet Protocol Version 6
+ User Datagram Protocol, Src Port: dhcpv6-client (546), Dst Port: dhcpv6-server (547)
− DHCPv6
    Message type: Solicit (1)
    Transaction-ID: 0x00817a48
  + Client Identifier
  + Identity Association for Non-temporary Address
  + Rapid Commit
  + Elapsed time
  − Option Request
      option type: 6
      option length: 4
      Requested Option code: DNS recursive name server (23)
      Requested Option code: Domain Search List (24)

0000  33 33 00 01 00 02 08 00  27 b0 13 5a 86 dd 60 00   33...... '..Z..`.
0010  00 00 00 40 11 40 fe 80  00 00 00 00 00 00 0a 00   ...@.@.. ........
0020  27 ff fe b0 13 5a ff 02  00 00 00 00 00 00 00 00   '....Z.. ........
0030  00 00 00 01 00 02 02 22  02 23 00 40 bf ef 01 81   ......." .#.@....
```

Figure 18 - DHCP Solicit

This Solicit message causes the DHCP server to Reply with the requested information, as shown in Figure 19.



```
No. .  Time          Source                       Destination                  Protocol  Info
   496 327.718991    fe80::a00:27ff:fe43:9ace     ff02::1:ffb0:135a            ICMPv6    Neighbor solicitation
   497 327.719117    fe80::a00:27ff:feb0:135a     fe80::a00:27ff:fe43:9ace     ICMPv6    Neighbor advertisement
   498 327.719231    fe80::a00:27ff:fe43:9ace     fe80::a00:27ff:feb0:135a     DHCPv6    Reply
   499 327.733681    fe80::a00:27ff:feb0:135a     ff02::16                     ICMPv6    Multicast Listener Report Message
   500 327.733694    fe80::a00:27ff:feb0:135a     ff02::16                     ICMPv6    Multicast Listener Report Message
   503 328.588333    ::                           ff02::1:ff00:3               ICMPv6    Neighbor solicitation
   504 328.588341    ::                           ff02::1:ff00:3               ICMPv6    Neighbor solicitation

        option type: 5
        option length: 24
        IPv6 address: 2003::3
        Preferred lifetime: 3600
        Valid lifetime: 18000
  + Rapid Commit
  − DNS recursive name server
      option type: 23
      option length: 16
      DNS servers address: 2003::3
  − Domain Search List
      option type: 24
      option length: 14
      DNS Domain Search List
      Domain: widetest.com

0090  46 50 00 0e 00 00 00 17  00 10 20 03 00 00 00 00   FP...... .. .....
00a0  00 00 00 00 00 00 00 00  00 0a 00 18 00 0e 08 77   ........ .......w
00b0  69 64 65 74 65 73 74 03  63 6f 6d 00               idetest. com.
```

Figure 19 - DHCP Reply

This process will result in the correct configuration of the host machine. However, this first test shows that a true “plug-and-play” mechanism is still not accomplished.

7.1.2 Dibbler

Dibbler was tested as an alternative to the previous daemon, because of its wider range of support for other operating systems. Although it is still necessary to install a client daemon in the host machine, *Dibbler* is available for, and was tested with, Linux, Windows XP, Windows Vista, and Windows Seven.

Dibbler is a portable DHCPv6 implementation. It supports stateful as well as stateless auto-configuration for IPv6. It is developed under the GNU GPL (General Public

License) license. It supports Server discovery and address assignment, (SOLICIT, ADVERTISE, REQUEST and REPLY messages), DNS and Domain Name information, NTP Servers, SIP Servers, as well as other options.

Installing the server daemon (Debian):

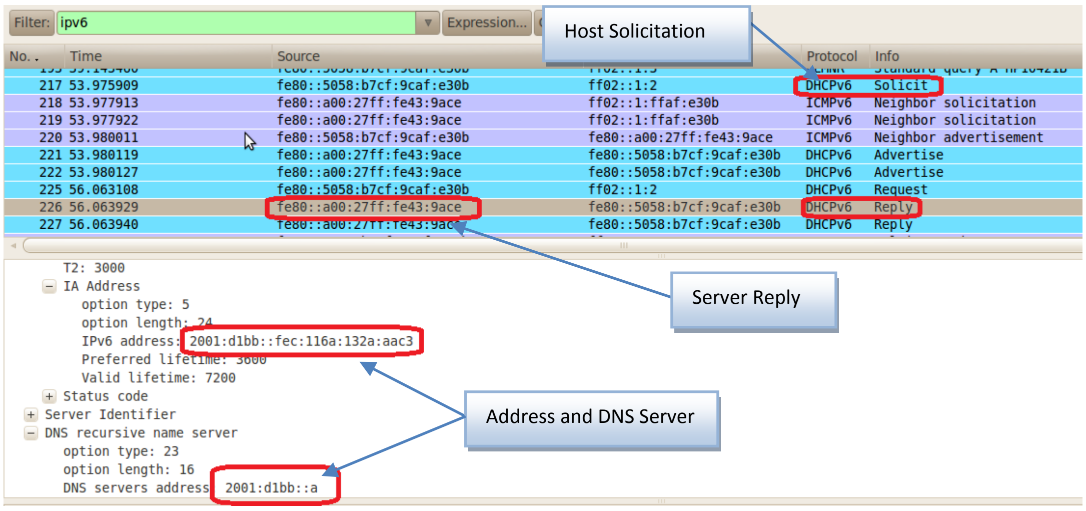
```
$apt-get install dibbler-server
```

The following modifications in Figure 20 were made to the *Dibbler* configuration file.

```
iface "eth0" {

 // also ranges can be defines, instead of exact values
 t1 1800-2000
 t2 2700-3000
 prefered-lifetime 3600
 valid-lifetime 7200

# assign addresses from this pool
 class {
   pool 2001:d1bb:/64
 }

# provide DNS server location to the clients
 option dns-server 2001:d1bb::a

# provide their domain name
 option domain dibblertest.com
```

Figure 20 - Configuration file server.conf

As with the previous daemon tested, the Solicit message is not sent by the host machine. Again, a client daemon (*Dibbler* or other) needs to be installed. Windows XP, Vista and Seven and Linux were all tested with this setup, to the same successful results, and, as it can be seen in Figure 21, the correct messages and options were exchanged:



```
No. .  Time          Source                        Destination                   Protocol  Info
217 53.975909  fe80::5058:b7cf:9caf:e30b  ff02::1:2                     DHCPv6    Solicit
218 53.977913  fe80::a00:27ff:fe43:9ace   ff02::1:ffaf:e30b             ICMPv6    Neighbor solicitation
219 53.977922  fe80::a00:27ff:fe43:9ace   ff02::1:ffaf:e30b             ICMPv6    Neighbor solicitation
220 53.980011  fe80::5058:b7cf:9caf:e30b  fe80::a00:27ff:fe43:9ace      ICMPv6    Neighbor advertisement
221 53.980119  fe80::a00:27ff:fe43:9ace   fe80::5058:b7cf:9caf:e30b     DHCPv6    Advertise
222 53.980127  fe80::a00:27ff:fe43:9ace   fe80::5058:b7cf:9caf:e30b     DHCPv6    Advertise
225 56.063108  fe80::5058:b7cf:9caf:e30b  ff02::1:2                     DHCPv6    Request
226 56.063929  fe80::a00:27ff:fe43:9ace   fe80::5058:b7cf:9caf:e30b     DHCPv6    Reply
227 56.063940  fe80::a00:27ff:fe43:9ace   fe80::5058:b7cf:9caf:e30b     DHCPv6    Reply

    T2: 3000
  − IA Address
      option type: 5
      option length: 24
      IPv6 address: 2001:d1bb::fec:116a:132a:aac3
      Preferred lifetime: 3600
      Valid lifetime: 7200
  + Status code
+ Server Identifier
− DNS recursive name server
    option type: 23
    option length: 16
    DNS servers address 2001:d1bb::a
```

Figure 21 – DHCP Message sequence and options using Dibbler

Still, and even though configuration was established with a wider range of OS's (Operative System), again the purpose of auto-configuration was defeated, leaving the end user with the responsibility of installing a DHCP client on his machine.

7.2 Stateless Configuration of Hosts

As mentioned in Chapter 6, stateless auto-configuration exploits several of the new features in IPv6, such as link-local addresses or the Neighbor Discovery protocol. It works by exchanging ICMP router discovery messages called "Router Advertisements" (RA) and "Router Solicitations" (RS). Each router periodically multicasts an RA from each of its multicast interfaces, announcing the IP address(es) of that interface. Hosts discover the addresses of their neighboring routers simply by listening for advertisements [20]. Figure 22 illustrates this message exchange:

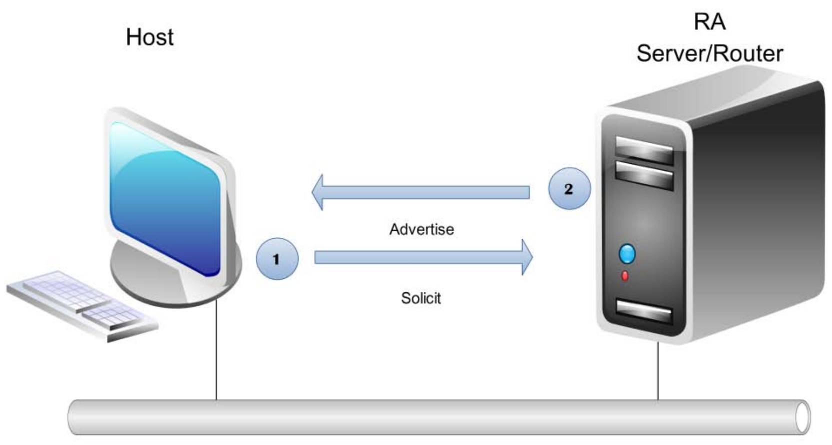


Figure 22 - Router Discovery process

RAs can be sent out from the routers themselves or from a server configured to act as a router. The latter was the approach taken in these tests. It should be noted that router advertisements do not explicitly announce the default gateway to the host, as it assumes that the source address of the RA is the actual network gateway (please refer to chapter 7.4 “The Default Router”). As such, a server running this service must have IPv6 Forwarding active.

In order to activate IPv6 forwarding the following entries were added to the */etc/sysctl.conf file.*

```
net.ipv6.conf.all.forwarding=1
net.ipv6.conf.default.forwarding=1
```

7.2.1 RADVD

RADVD (Router Advertisement Daemon) is an open-source software that implements link-local advertisements of IPv6 router addresses and IPv6 routing prefixes using the Neighbor Discovery Protocol (NDP) as specified in RFC 2461.

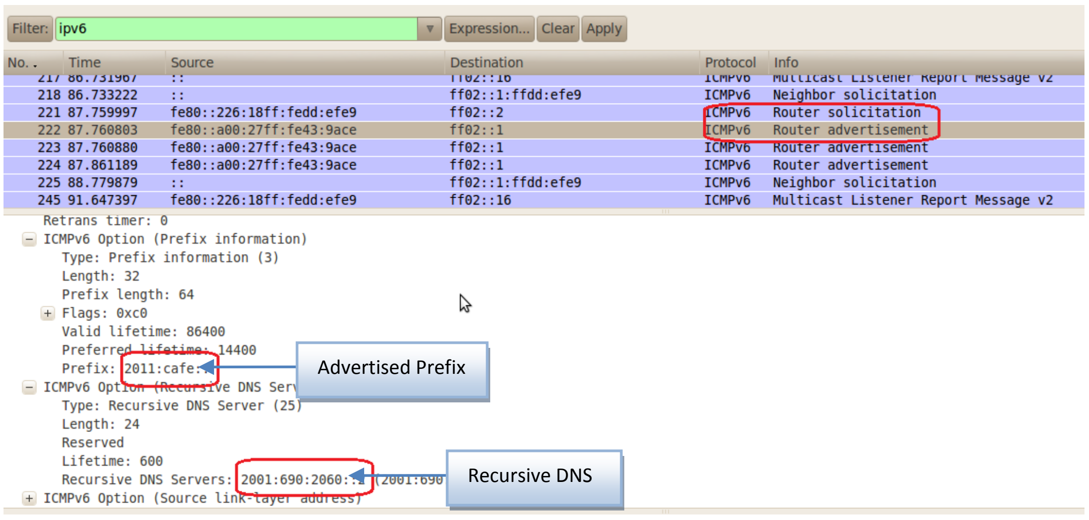
The daemon is configured by editing the file "*/etc/radvd.conf*". Figure 23 exemplifies configurations that were performed.

```
interface eth0{
     AdvSendAdvert on;
     AdvManagedFlag off;
     AdvOtherConfigFlag off;
     AdvDefaultPreference high;
     prefix 2011:cafe::/64
     {
          AdvOnLink on;
          AdvAutonomous on;
     };
     RDNSS 2001:690:2060::2
     {
        AdvRDNSSPreference 8;
       AdvRDNSSOpen on;

     };
}
```

Figure 23 - Configuration file radvd.conf

Both flags ("OtherConfig" and "Managed") were deactivated, since the purpose of this test is to get a true stateless auto-configuration.

According to Figure 24, a Router Solicitation message is sent from the host machine to the well-known multicast address "ff02::2" (All Routers), which provokes a RA message, containing the network prefix and RDDNS (Recursive DNS) information:



```
No. .  Time         Source                      Destination           Protocol  Info
  217 86.731967    ::                          ff02::16              ICMPv6    Multicast Listener Report Message v2
  218 86.733222    ::                          ff02::1:ffdd:efe9     ICMPv6    Neighbor solicitation
  221 87.759997    fe80::226:18ff:fedd:efe9    ff02::2               ICMPv6    Router solicitation
  222 87.760803    fe80::a00:27ff:fe43:9ace    ff02::1               ICMPv6    Router advertisement
  223 87.760880    fe80::a00:27ff:fe43:9ace    ff02::1               ICMPv6    Router advertisement
  224 87.861189    fe80::a00:27ff:fe43:9ace    ff02::1               ICMPv6    Router advertisement
  225 88.779879    ::                          ff02::1:ffdd:efe9     ICMPv6    Neighbor solicitation
  245 91.647397    fe80::226:18ff:fedd:efe9    ff02::16              ICMPv6    Multicast Listener Report Message v2

    Retrans timer: 0
  − ICMPv6 Option (Prefix information)
      Type: Prefix information (3)
      Length: 32
      Prefix length: 64
    + Flags: 0xc0
      Valid lifetime: 86400
      Preferred lifetime: 14400
      Prefix: 2011:cafe::
  − ICMPv6 Option (Recursive DNS Serv
      Type: Recursive DNS Server (25)
      Length: 24
      Reserved
      Lifetime: 600
      Recursive DNS Servers: 2001:690:2060::2 (2001:690
  + ICMPv6 Option (Source link-layer address)
```

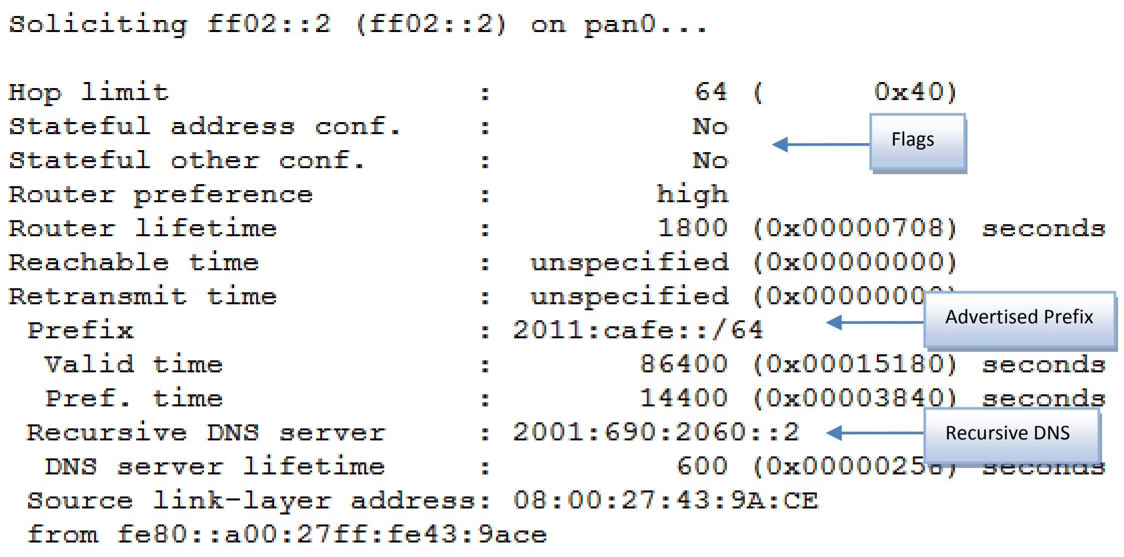
Figure 24 - Router Advertisement

The host machines configure themselves using the network prefix advertised, and an Enhanced Unique Identifier, EUI-64, generated from its interface's MAC address.

This process can also be seen using *ndisc6*, a group of diagnostic tools for Linux and BSD. It includes *rdisc6*, an ICMPv6 Router Discovery tool which queries IPv6 routers on the network for advertised prefixes. It can be used to detect rogue IPv6 routers and monitor legitimate IPv6 routers [21];

```
$ rdisc6 interface
```

This tool shows us all RA information running on a specific interface (*pan0* in this case):



```
Soliciting ff02::2 (ff02::2) on pan0...

Hop limit                 :           64 (      0x40)
Stateful address conf.    :           No
Stateful other conf.      :           No
Router preference         :         high
Router lifetime           :         1800 (0x00000708) seconds
Reachable time            :  unspecified (0x00000000)
Retransmit time           :  unspecified (0x0000000
 Prefix                   : 2011:cafe::/64
  Valid time              :        86400 (0x00015180) seconds
  Pref. time              :        14400 (0x00003840) seconds
 Recursive DNS server     : 2001:690:2060::2
  DNS server lifetime     :          600 (0x0000025
 Source link-layer address: 08:00:27:43:9A:CE
 from fe80::a00:27ff:fe43:9ace
```

Figure 25 - *rdisc6* output

All OS's tested with this method (except Windows Mobile 6.1) obtained a valid network address. This was especially encouraging since none of the other methods worked on the Android platform[2].

The resulting problem in this approach is that the RDDNS information is not interpreted by most of the OS's, which simply disregard it. In fact, only in the latest builds of Ubuntu (11.04) and Fedora (15), both an address and a DNS entry were configured automatically. Hosts running Ubuntu 10.04 LTS needed an extra daemon installed in order to process this information, called *rddns*. In Windows, we were also able to bypass this issue by installing a daemon called '*RDNSS_win32*', which is an implementation of the RFC 6106 for Microsoft Windows. It parses the RDNSS options present in Router Advertisement (RA), gets the Name Servers and writes them into the registry, which was not considered to be an acceptable solution, since those entries cannot be flushed, and have to be manually removed.

[2] Refer to sub-chapter 1.1.4, as wireless connectivity achieved is merely momentary

7.3 Combining Stateless with Stateful

The next logical step was combining both processes, and trying to use each of their strengths.

By introducing RAs, host machines not running any specific DHCPv6 client can be induced into requesting additional options such as DNS to a DHCPv6 server:

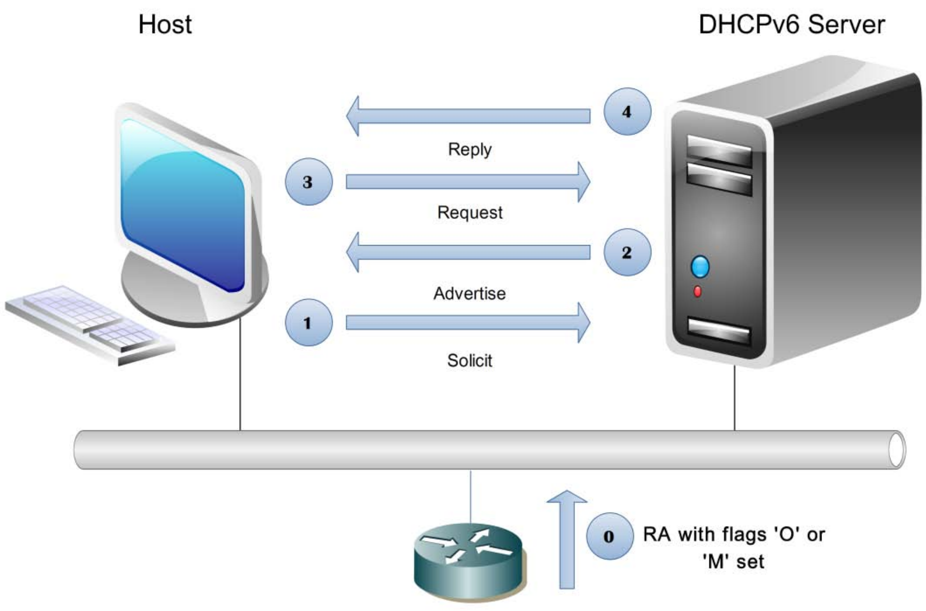


Figure 26 - Introducing RA to DHCP Message Exchange

The addition of step 0 in Figure 26 shows the introduction of a flagged RA in the process. Two groups of tests were made, with one or the other of the flags set.

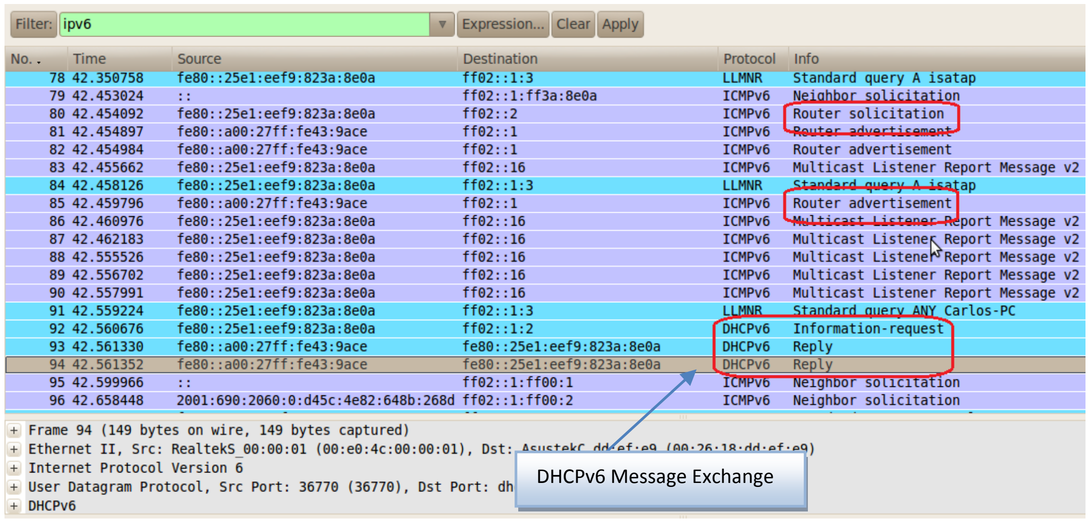
- ‘O’ Flag Set

Setting the ‘M’ bit to zero and the ‘O’ bit to one will cause host machines to get the network prefix from the RA (using EUI-64 to set the host part of the address), and additional information from a DHCPv6 Server.

The following entries were modified in the *radvd* configuration file:

```
AdvManagedFlag off;
AdvOtherConfigFlag on;
```

Figure 27 shows the sequence of messages that were triggered successfully.



```
No.  Time       Source                                 Destination              Protocol  Info
 78 42.350758   fe80::25e1:eef9:823a:8e0a              ff02::1:3                LLMNR     Standard query A isatap
 79 42.453024   ::                                     ff02::1:ff3a:8e0a        ICMPv6    Neighbor solicitation
 80 42.454092   fe80::25e1:eef9:823a:8e0a              ff02::2                  ICMPv6    Router solicitation
 81 42.454897   fe80::a00:27ff:fe43:9ace               ff02::1                  ICMPv6    Router advertisement
 82 42.454984   fe80::a00:27ff:fe43:9ace               ff02::1                  ICMPv6    Router advertisement
 83 42.455662   fe80::25e1:eef9:823a:8e0a              ff02::16                 ICMPv6    Multicast Listener Report Message v2
 84 42.458126   fe80::25e1:eef9:823a:8e0a              ff02::1:3                LLMNR     Standard query A isatap
 85 42.459796   fe80::a00:27ff:fe43:9ace               ff02::1                  ICMPv6    Router advertisement
 86 42.460976   fe80::25e1:eef9:823a:8e0a              ff02::16                 ICMPv6    Multicast Listener Report Message v2
 87 42.462183   fe80::25e1:eef9:823a:8e0a              ff02::16                 ICMPv6    Multicast Listener Report Message v2
 88 42.555526   fe80::25e1:eef9:823a:8e0a              ff02::16                 ICMPv6    Multicast Listener Report Message v2
 89 42.556702   fe80::25e1:eef9:823a:8e0a              ff02::16                 ICMPv6    Multicast Listener Report Message v2
 90 42.557991   fe80::25e1:eef9:823a:8e0a              ff02::16                 ICMPv6    Multicast Listener Report Message v2
 91 42.559224   fe80::25e1:eef9:823a:8e0a              ff02::1:3                LLMNR     Standard query ANY Carlos-PC
 92 42.560676   fe80::25e1:eef9:823a:8e0a              ff02::1:2                DHCPv6    Information-request
 93 42.561330   fe80::a00:27ff:fe43:9ace               fe80::25e1:eef9:823a:8e0a DHCPv6   Reply
 94 42.561352   fe80::a00:27ff:fe43:9ace               fe80::25e1:eef9:823a:8e0a DHCPv6   Reply
 95 42.599966   ::                                     ff02::1:ff00:1           ICMPv6    Neighbor solicitation
 96 42.658448   2001:690:2060:0:d45c:4e82:648b:268d    ff02::1:ff00:2           ICMPv6    Neighbor solicitation

+ Frame 94 (149 bytes on wire, 149 bytes captured)
+ Ethernet II, Src: RealtekS_00:00:01 (00:e0:4c:00:00:01), Dst: AsustekC_dd:ef:e9 (00:26:18:dd:ef:e9)
+ Internet Protocol Version 6
+ User Datagram Protocol, Src Port: 36770 (36770), Dst Port: dh
+ DHCPv6
```

Figure 27 - RA triggered DHCP Request and Reply

This method was proven to be successful with most OS's, especially Windows Vista, Seven, and Ubuntu 11.04. These hosts acquired the prefix, created the EUI-64 address, and set correct DNS parameters that originated from the DHCP server. The remaining hosts set the address, but failed to set the DNS entries.

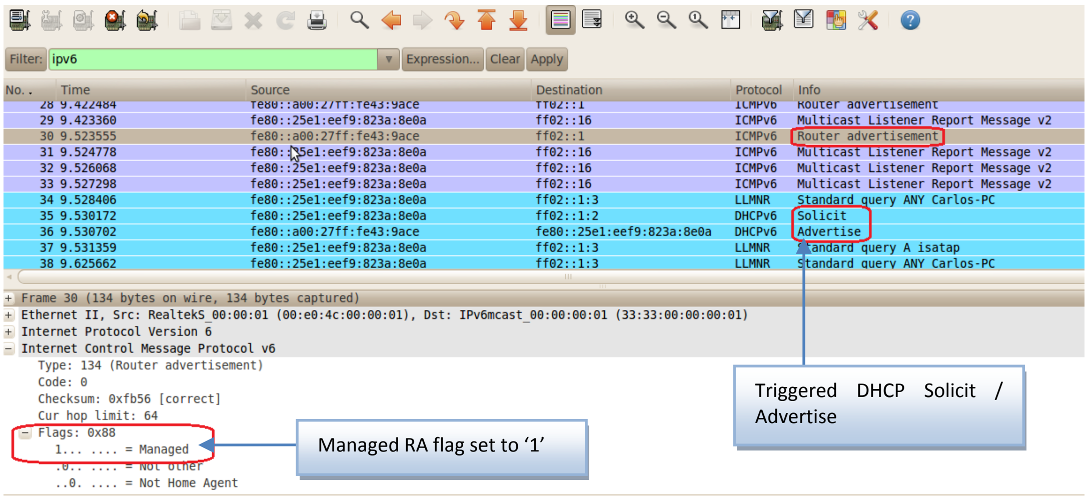
- 'M' Flag Set

By setting the Managed flag to one, hosts are "instructed" to obtain all relative information from the DHCP server, with the added functionality that if assignment of IP through DHCP fails, it will still store a temporary EUI-64 global address.

The *radvd* configuration file was modified in the following entries:

```
AdvManagedFlag on;
AdvOtherConfigFlag off;
```

In Figure 28 it is possible to see the flag set-up, as well as the triggered DHCP response.



```
No. .  Time        Source                       Destination                  Protocol  Info
  28 9.422484   fe80::a00:27ff:fe43:9ace     ff02::1                      ICMPv6    Router advertisement
  29 9.423360   fe80::25e1:eef9:823a:8e0a    ff02::16                     ICMPv6    Multicast Listener Report Message v2
  30 9.523555   fe80::a00:27ff:fe43:9ace     ff02::1                      ICMPv6    Router advertisement
  31 9.524778   fe80::25e1:eef9:823a:8e0a    ff02::16                     ICMPv6    Multicast Listener Report Message v2
  32 9.526068   fe80::25e1:eef9:823a:8e0a    ff02::16                     ICMPv6    Multicast Listener Report Message v2
  33 9.527298   fe80::25e1:eef9:823a:8e0a    ff02::16                     ICMPv6    Multicast Listener Report Message v2
  34 9.528406   fe80::25e1:eef9:823a:8e0a    ff02::1:3                    LLMNR     Standard query ANY Carlos-PC
  35 9.530172   fe80::25e1:eef9:823a:8e0a    ff02::1:2                    DHCPv6    Solicit
  36 9.530702   fe80::a00:27ff:fe43:9ace     fe80::25e1:eef9:823a:8e0a    DHCPv6    Advertise
  37 9.531359   fe80::25e1:eef9:823a:8e0a    ff02::1:3                    LLMNR     Standard query A isatap
  38 9.625662   fe80::25e1:eef9:823a:8e0a    ff02::1:3                    LLMNR     Standard query ANY Carlos-PC

+ Frame 30 (134 bytes on wire, 134 bytes captured)
+ Ethernet II, Src: RealtekS_00:00:01 (00:e0:4c:00:00:01), Dst: IPv6mcast_00:00:00:01 (33:33:00:00:00:01)
+ Internet Protocol Version 6
- Internet Control Message Protocol v6
    Type: 134 (Router advertisement)
    Code: 0
    Checksum: 0xfb56 [correct]
    Cur hop limit: 64
  - Flags: 0x88
      1... .... = Managed
      .0.. .... = Not other
      ..0. .... = Not Home Agent
```

Managed RA flag set to ‘1’
Triggered DHCP Solicit / Advertise

Figure 28 - RA with 'M' Flag

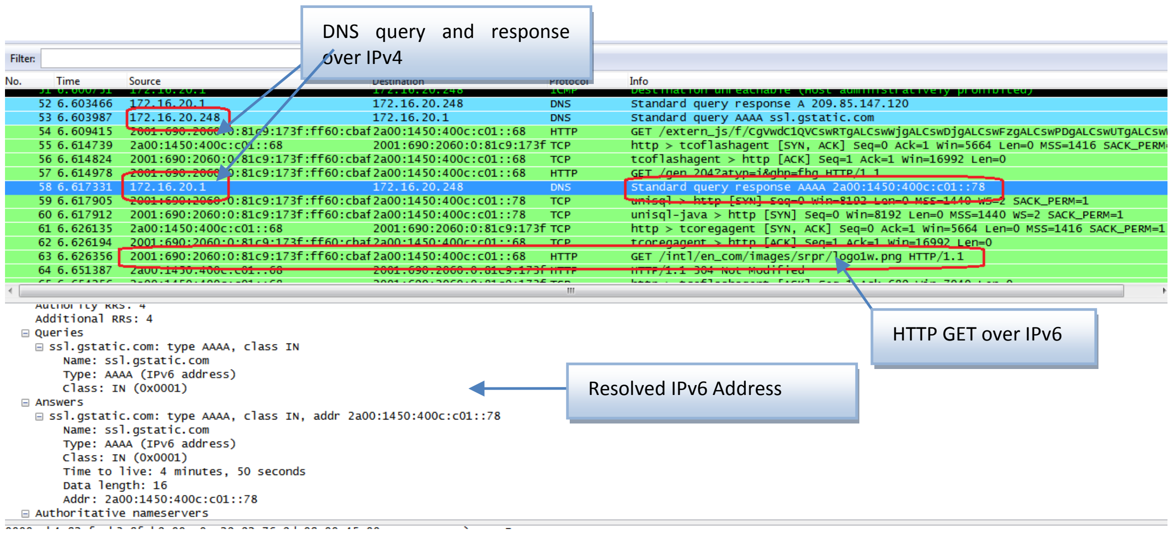
This method proved to be interesting because, besides the fall-back mentioned earlier, Ubuntu 11.04, Fedora 15, Windows Vista and Seven, all set the appropriate configurations sent through DHCP.

7.4 The Dual Stack Method

Having tested the different methods IPv6 has to offer, one additional solution was tested. Since some of the OS’s tested failed consistently at obtaining the additional (and vital) options such as DNS, we chose to combine a stateless auto-configuration method for addressing, with an IPv4 DHCP server for name resolution. We also came to the conclusion that one of the OS’s tested (Android) failed to maintain connectivity without having an IPv4 address assigned, so we set up the same stateless scenario with *radvd* (all flags off) and tested to see if connectivity was established over IPv6, even though the name resolution was over IPv4.

Testing at this phase was done on a newly available native IPv6 connection to FCCN (Foundation for National Scientific Computing) and, consequently, a new scenario was created. For details and network diagram, please refer to chapter “World IPv6 Day”.

As can be seen in Figure 29, and, as expected, although DNS queries and replies are all done through IPv4, resolution of ‘AAAA’ records still provides the correct IPv6 addresses through which an IPv6 network can be accessed properly.



```
No.   Time      Source                                Destination                          Protocol  Info
   51 6.600731  172.16.20.1                           172.16.20.248                        ICMP      Destination unreachable (Host administratively prohibited)
   52 6.603466  172.16.20.1                           172.16.20.248                        DNS       Standard query response A 209.85.147.120
   53 6.603987  172.16.20.248                         172.16.20.1                          DNS       Standard query AAAA ssl.gstatic.com
   54 6.609415  2001:690:2060:0:81c9:173f:ff60:cbaf   2a00:1450:400c:c01::68               HTTP      GET /extern_js/f/CgVwdC1QVCswRTgALCswWjgALCswDjgALCswFzgALCswPDgALCswUTgALCsw
   55 6.614739  2a00:1450:400c:c01::68                2001:690:2060:0:81c9:173f            TCP       http > tcoflashagent [SYN, ACK] Seq=0 Ack=1 Win=5664 Len=0 MSS=1416 SACK_PERM=
   56 6.614824  2001:690:2060:0:81c9:173f:ff60:cbaf   2a00:1450:400c:c01::68               TCP       tcoflashagent > http [ACK] Seq=1 Ack=1 Win=16992 Len=0
   57 6.614978  2001:690:2060:0:81c9:173f:ff60:cbaf   2a00:1450:400c:c01::68               HTTP      GET /gen_204?atyp=i&ghp=fbg HTTP/1.1
   58 6.617331  172.16.20.1                           172.16.20.248                        DNS       Standard query response AAAA 2a00:1450:400c:c01::78
   59 6.617905  2001:690:2060:0:81c9:173f:ff60:cbaf   2a00:1450:400c:c01::78               TCP       unisql > http [SYN] Seq=0 Win=8192 Len=0 MSS=1440 WS=2 SACK_PERM=1
   60 6.617912  2001:690:2060:0:81c9:173f:ff60:cbaf   2a00:1450:400c:c01::78               TCP       unisql-java > http [SYN] Seq=0 Win=8192 Len=0 MSS=1440 WS=2 SACK_PERM=1
   61 6.626135  2a00:1450:400c:c01::68                2001:690:2060:0:81c9:173f            TCP       http > tcoregagent [SYN, ACK] Seq=0 Ack=1 Win=5664 Len=0 MSS=1416 SACK_PERM=1
   62 6.626194  2001:690:2060:0:81c9:173f:ff60:cbaf   2a00:1450:400c:c01::68               TCP       tcoregagent > http [ACK] Seq=1 Ack=1 Win=16992 Len=0
   63 6.626356  2001:690:2060:0:81c9:173f:ff60:cbaf   2a00:1450:400c:c01::68               HTTP      GET /intl/en_com/images/srpr/logo1w.png HTTP/1.1
   64 6.651387  2a00:1450:400c:c01::68                2001:690:2060:0:81c9:173f            HTTP      HTTP/1.1 304 Not Modified
```

```
    Authority RRs: 4
    Additional RRs: 4
  ⊟ Queries
    ⊟ ssl.gstatic.com: type AAAA, class IN
        Name: ssl.gstatic.com
        Type: AAAA (IPv6 address)
        Class: IN (0x0001)
  ⊟ Answers
    ⊟ ssl.gstatic.com: type AAAA, class IN, addr 2a00:1450:400c:c01::78
        Name: ssl.gstatic.com
        Type: AAAA (IPv6 address)
        Class: IN (0x0001)
        Time to live: 4 minutes, 50 seconds
        Data length: 16
        Addr: 2a00:1450:400c:c01::78
  ⊟ Authoritative nameservers
```

Figure 29 - IPv6 Connectivity with IPv4 DNS queries

This method worked flawlessly for all IPv6-capable systems tested. It allows for configuration with no user intervention, and overcomes the problem encountered in Android systems not being capable of maintaining connectivity with no IPv4 address assigned.

7.5 The Default Router

As stated previously, all these tests were performed using a server as the source of the Router Advertisements. Since the host machines assume that the node supplying the RA is the default gateway, the first hop for all outward traffic is always the server (Figure 30).

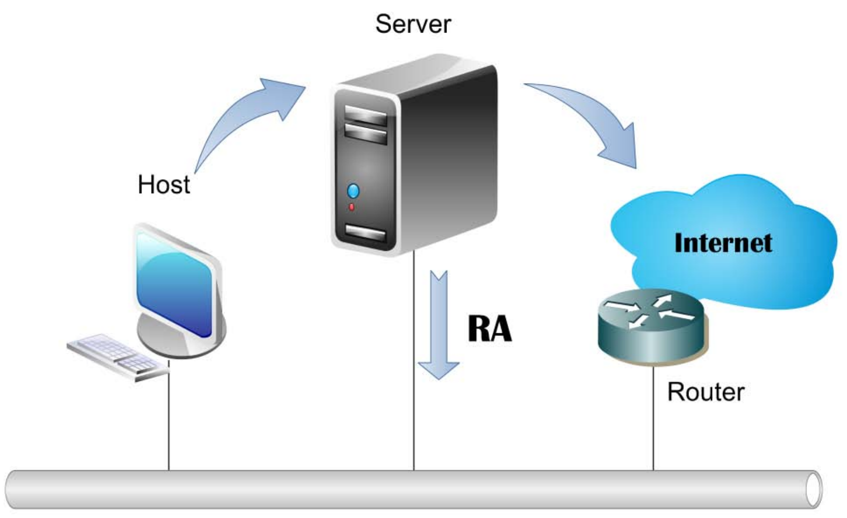


Figure 30 - Traffic path with server-originated RA

This process can become troublesome in larger networks, as every packet needs to be forwarded by the server. There are two solutions to this problem: the RA's can originate from the outward router itself with the desired parameters (i.e. flags) and exist as the only source of advertisements on the link, or the server's *Router Lifetime* and *Default Preference* parameters can be manipulated.

The router lifetime is possibly a slightly misnamed field. A *Lifetime* of 0 indicates that the router is not a default router and should not appear on the default router list. The *Router Lifetime* applies only to the router's usefulness as a default router; it does not apply to information contained in other message fields or options [23].

The *Default Preference* parameter specifies which of the circulating RAs the host should use for stateless configuration.

Figure 31 shows both RAs being introduced into the link. The server RA contains the active flags, with a high preference and a lifetime of zero, indicating that it should not be used as default router. The router RA has no flags, a lower preference, and a bigger than zero lifetime.

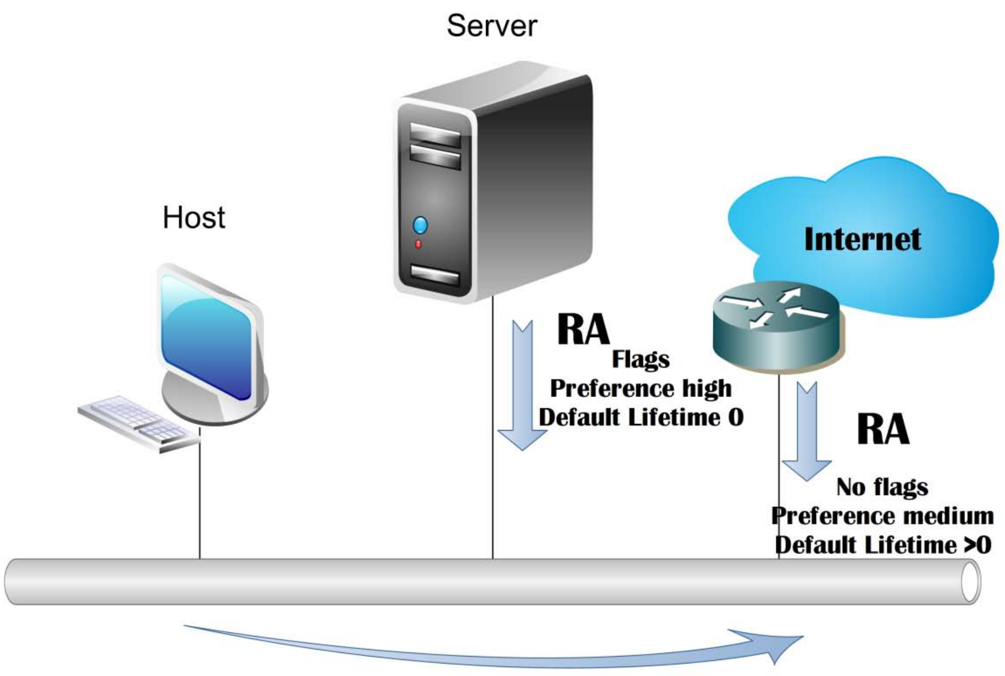
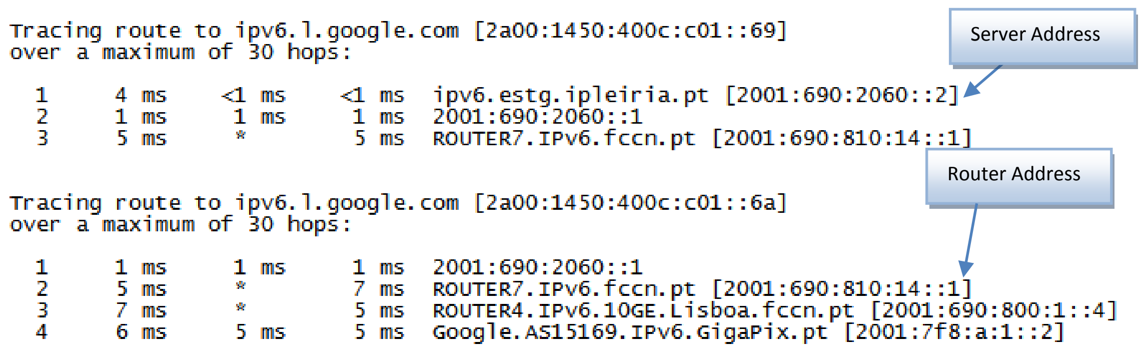


Figure 31 - Traffic path with both server and router originated RAs

Figure 32 shows resulting *traceroute* outputs, before and after the preference and lifetime were set.



```
Tracing route to ipv6.l.google.com [2a00:1450:400c:c01::69]
over a maximum of 30 hops:

  1     4 ms    <1 ms    <1 ms  ipv6.estg.ipleiria.pt [2001:690:2060::2]
  2     1 ms     1 ms     1 ms  2001:690:2060::1
  3     5 ms     *        5 ms  ROUTER7.IPv6.fccn.pt [2001:690:810:14::1]

Tracing route to ipv6.l.google.com [2a00:1450:400c:c01::6a]
over a maximum of 30 hops:

  1     1 ms     1 ms     1 ms  2001:690:2060::1
  2     5 ms     *        7 ms  ROUTER7.IPv6.fccn.pt [2001:690:810:14::1]
  3     7 ms     *        5 ms  ROUTER4.IPv6.10GE.Lisboa.fccn.pt [2001:690:800:1::4]
  4     6 ms     5 ms     5 ms  Google.AS15169.IPv6.GigaPix.pt [2001:7f8:a:1::2]
```

Figure 32 - Traffic path (traceroute output)

Notice that in the first *traceroute*, traffic is forwarded by the server (2001:690:2060::2) while in the second one, the host ignored the server as default router and, therefore, used the router as first hop.

As for having the router itself issue all necessary parameters in the RA, the following set of configurations can be applied:

```
(config-if)#ipv6 nd prefix 2001:690:2060::/64
(config-if)#ipv6 nd other-config-flag
(config-if)#ipv6 nd advertisement-interval 60
(config-if)#ipv6 nd ra lifetime 1800
(config-if)#ipv6 nd router-preference high
```

7.6 Conclusions

After having run all the tests, we came to the conclusion that older operating systems offer some difficulty which could cause significant connectivity problems in a diverse network. Windows XP was one of these cases, as it failed in all tests that did not include manually installing a DHCPv6 client. However, much recent OS's such as Android also showed similar problems.

Table 7 shows all results achieved in this chapter, regarding IPv6 compatibility and readiness. Note that for an OS to be considered compatible in any given method must have been able to achieve full network connectivity without user intervention (i.e. manually installing a daemon).

Table 7 - Operating Systems IPv6 Compatibility

	DHCPv6 (Stateful)	RA (Stateless)	'Other' flag set	'Managed' flag set	RA with DHCPv4
Android	No	Address Only	Address Only	Address Only	Yes
Windows Mobile 6.1	No	No	No	No	No
Ubuntu 11.04	Requires third party client	Yes	Yes	Yes	Yes
Ubuntu 10.04 LTS	Requires third party client	Address Only	Address Only	Address Only	Yes
Fedora 15	Requires third party client	Yes	Address Only	Yes	Yes
Windows Seven	Requires third party client	Address Only	Yes	Yes	Yes
Windows Vista	Requires third party client	Address Only	Yes	Yes	Yes
Windows XP	Requires third party client	Address Only	Address Only	Address Only	Yes
MacOS	Requires third party client	Address Only	Address Only	Address Only	Yes

Tests that resulted in "Address Only" being achieved meant that the OS either ignored the RA flags, or failed to interpret Recursive DNS Server options. The host, even though instructed to get an address statefully, acquired a stateless EUI-64 address only.

In fact, according to RFC 5006, it is to be expected that in environments where RDNSS information is stored in user space and ND runs in the kernel, such as Ubuntu 10.04 and Windows Seven for instance, a daemon must be used to monitor RDNSS addresses.

In times of transition, when both protocols are expected to run side by side for a long time yet, the final approach seems to be the better one. Name resolution through IPv4 solves many of these problems, including Android's inability to maintain an IPv6-only network connection.

Recent distributions such as Fedora 15 and Ubuntu 11.04 performed remarkably well, being able to achieve a full stateless auto-configuration. Using flagged Router Advertisements in conjunction with DHCPv6 for additional options proved to be a quite successful combination with Microsoft’s Operating Systems.

Regarding the fact of the default router, ideally the only router advertisement should originate from the router itself. If not possible, and a server is used, the method described was tested as a viable option, and should be taken into consideration.

Chapter 8

Authentication & Security Tests

In order to test the security mechanisms, we implemented some test scenarios using RADIUS protocol (authentication 802.1X). There are also some hardware (Access point and Switch) configurations, which can protect and secure a network and prevent from attacks, which is what we analyzed in the last part of this chapter.

The testbed used for these tests is as in Figure 33.

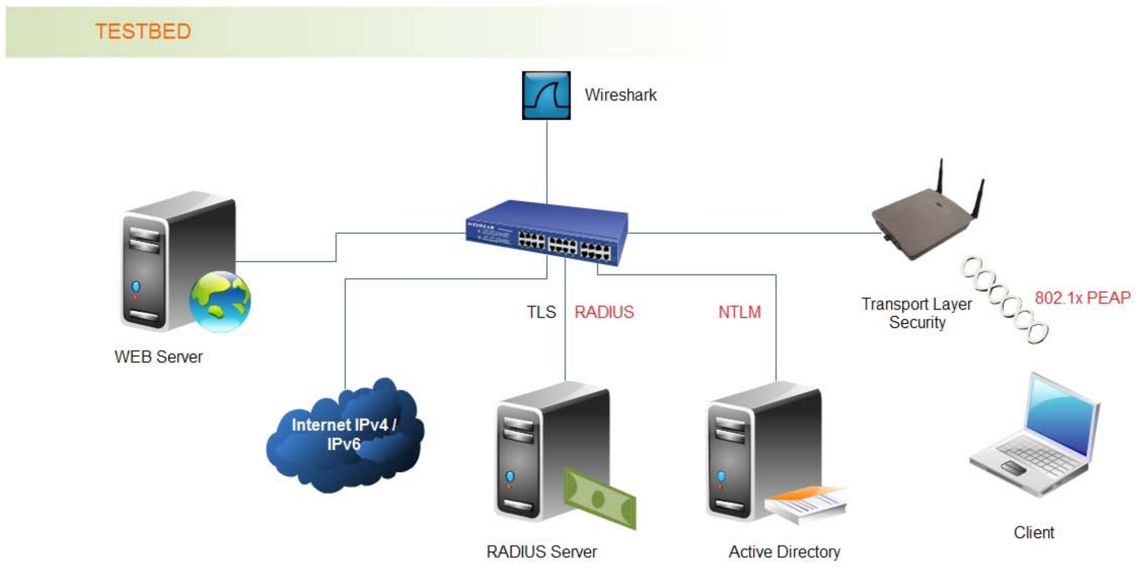


Figure 33 – Authentication & Security testbed

8.1 RADIUS Tests

Four tests were implemented with RADIUS. The first one involves Active Directory integration with IPv4, the second is a test scenario with NPS (Network Policy Server) in Windows Server 2008 also with IPv4, and finally we tested the IPv6 compatibility in a Wi-Fi and port based (wired) scenario, using RADIUS with Active Directory.

8.1.1 RADIUS with Active Directory integration

Our first test scenario consisted in deploying RADIUS with Active directory. In this case the validation of the user/client credentials is done through the active directory, which acts as a repository of users' credentials in a domain. This mechanism involves information exchange between a RADIUS server and the Active directory. To do so, we used an open source version of RADIUS called FreeRADIUS, and the OS platform chosen was CentOS 5.5.

Regarding the Active Directory (AD), we used Windows Server 2008. Figure 34 exhibits this scenario.

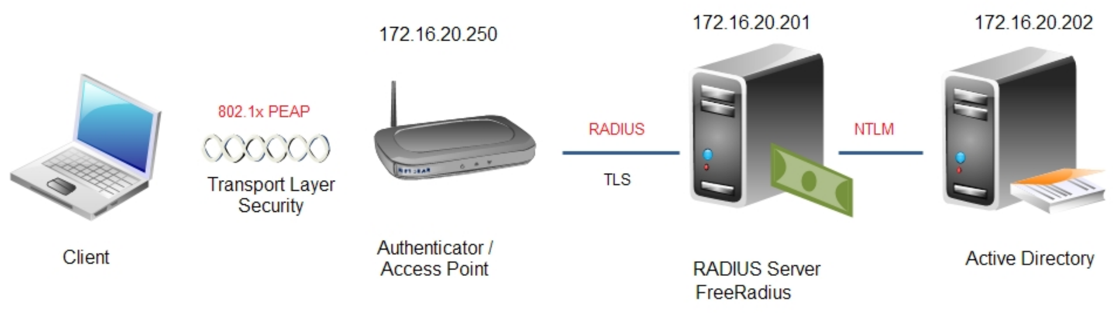


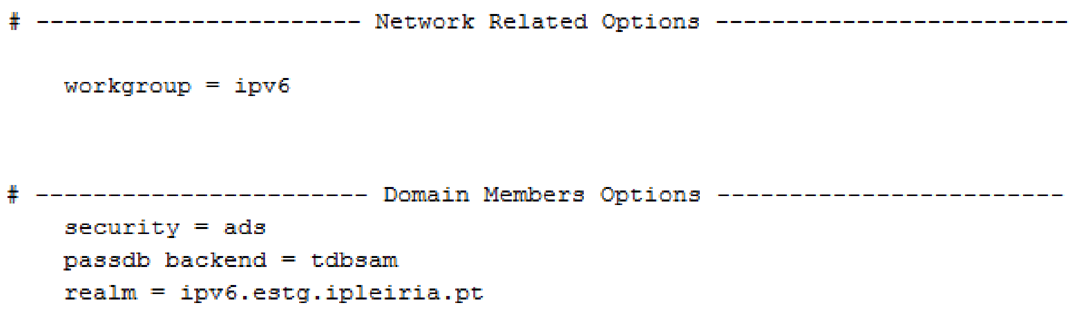
Figure 34 – RADIUS with AD integration

In this scenario, we implemented and tested the authentication process with IPv4, where the client will try to connect to the network (SSID=IPv6) in a Dual Stack environment.

Following components were used: linux Server, FreeRADIUS, Samba 3.0.x, Openssl, an access point with WPA enterprise support and a Cisco switch catalyst.

Before installing and configuring FreeRADIUS, it is necessary to configure Samba and Kerberos services and to set the winbind service in the /etc/*nsswitch.conf* file.

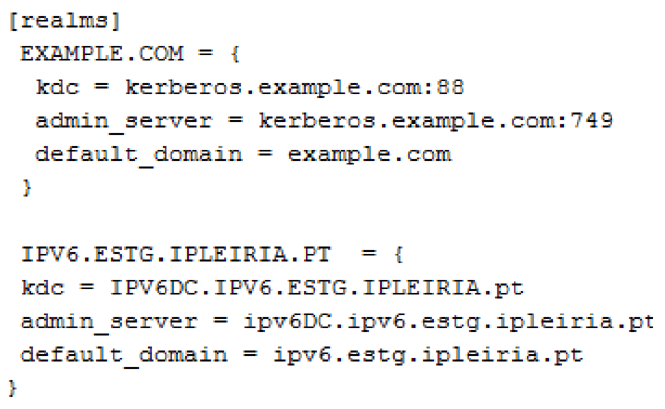
In the samba configuration file */etc/samba/smb.conf* (Figure 35), it is necessary to edit the Global and the Domain Member section.



```
# ----------------------- Network Related Options -------------------------

    workgroup = ipv6


# ----------------------- Domain Members Options ------------------------
    security = ads
    passdb backend = tdbsam
    realm = ipv6.estg.ipleiria.pt
```

Figure 35 - Configuration file smb.conf

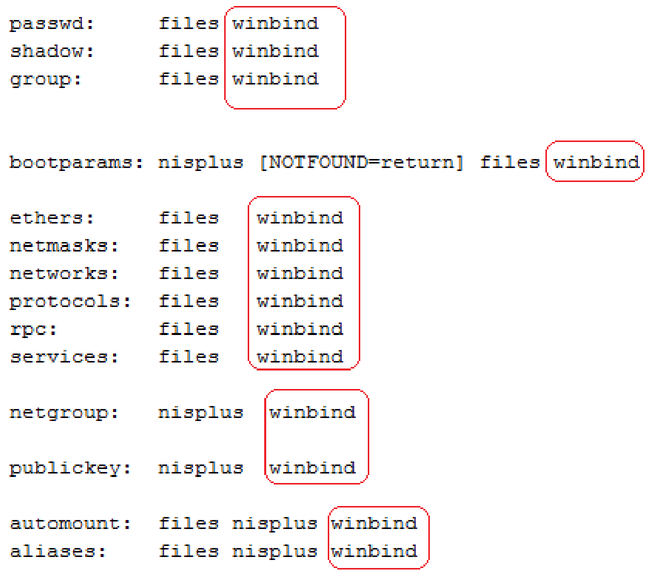
It is necessary to add also an entry in the Kerberos configuration file (*etc/krb5.conf*) in order to point to our Active directory, as shown in Figure 36.



```
[realms]
 EXAMPLE.COM = {
  kdc = kerberos.example.com:88
  admin_server = kerberos.example.com:749
  default_domain = example.com
 }

 IPV6.ESTG.IPLEIRIA.PT  = {
 kdc = IPV6DC.IPV6.ESTG.IPLEIRIA.pt
 admin_server = ipv6DC.ipv6.estg.ipleiria.pt
 default_domain = ipv6.estg.ipleiria.pt
}
```

Figure 36 - Configuration file krb5.conf

Then, the *nsswitch.conf* file must be edited with the *winbind* value in the sections marked in figure Figure 37.



```
passwd:     files winbind
shadow:     files winbind
group:      files winbind

bootparams: nisplus [NOTFOUND=return] files winbind

ethers:     files   winbind
netmasks:   files   winbind
networks:   files   winbind
protocols:  files   winbind
rpc:        files   winbind
services:   files   winbind

netgroup:   nisplus  winbind

publickey:  nisplus  winbind

automount:  files nisplus winbind
aliases:    files nisplus winbind
```

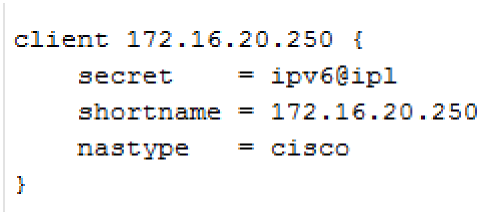
Figure 37 –Configuration file *nsswitch.conf*

To install FreeRADIUS in CentOS, we run following command:

yum install Freeradius x.x.x.x

where x.x.x.x stands for the last current version number.

The configuration of FreeRADIUS involves initially following files located in directories */etc/raddb* and */etc/raddb/modules*: Clients.conf, Eap.conf, ntlm_auth and mschap.



```
client 172.16.20.250 {
     secret     = ipv6@ipl
     shortname = 172.16.20.250
     nastype    = cisco
}
```

Figure 38 – Configuration file *clients.conf*

The *clients.conf* file (Figure 38) defines the RADIUS client access control.

The secret field is the common password between the Access Point and the RADIUS server. It prevents the installation of wilds access points.

The shortname field is the IP address of the Access Point

The nastype indicates the type of Access Point. In our case, this is a Cisco Access Point.

Now regarding the *eap.conf* file, we specify the authentication method used by FreeRADIUS. We chose PEAP (Protected EAP), as it allows using MSCHAPv2 a challenge/response protocol to authenticate against an Active Directory Windows Domain.

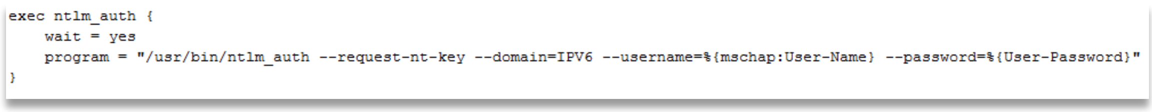
```
default_eap_type = peap
```

In order to communicate with the Active Directory, FreeRADIUS will use following components: Winbind and Ntlm_auth.

Winbind is a daemon with permits connectivity to Windows NT environment.

Ntlm_auth is a tool which uses winbind for evaluating NTLM (NT Lan Manager) requests. This tool allows verifying user credentials on the domain controller and returns either a success or an error message).

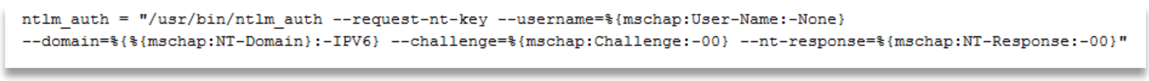
Figure 39 shows configuration for ntlm_auth.



```
exec ntlm_auth {
    wait = yes
    program = "/usr/bin/ntlm_auth --request-nt-key --domain=IPV6 --username=%{mschap:User-Name} --password=%{User-Password}"
}
```

Figure 39 - ntlm_auth

To get FreeRADIUS to use ntlm_auth for MS-CHAP, the *mschap file* must be edited as exemplified in figure Figure 40.

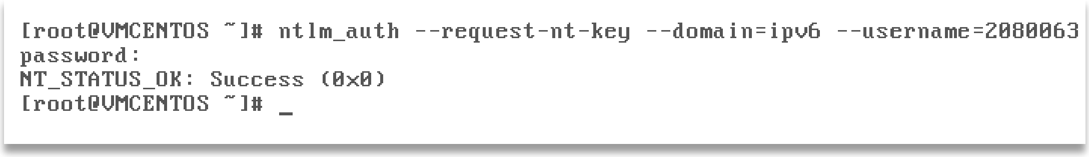


```
ntlm_auth = "/usr/bin/ntlm_auth --request-nt-key --username=%{mschap:User-Name:-None}
--domain=%{%{mschap:NT-Domain}:-IPV6} --challenge=%{mschap:Challenge:-00} --nt-response=%{mschap:NT-Response:-00}"
```

Figure 40 - Configuration file mschap

Regarding the tests for this scenario, we used the tools mentioned before (winbind and ntlm_auth) to test the authentication mechanism locally.

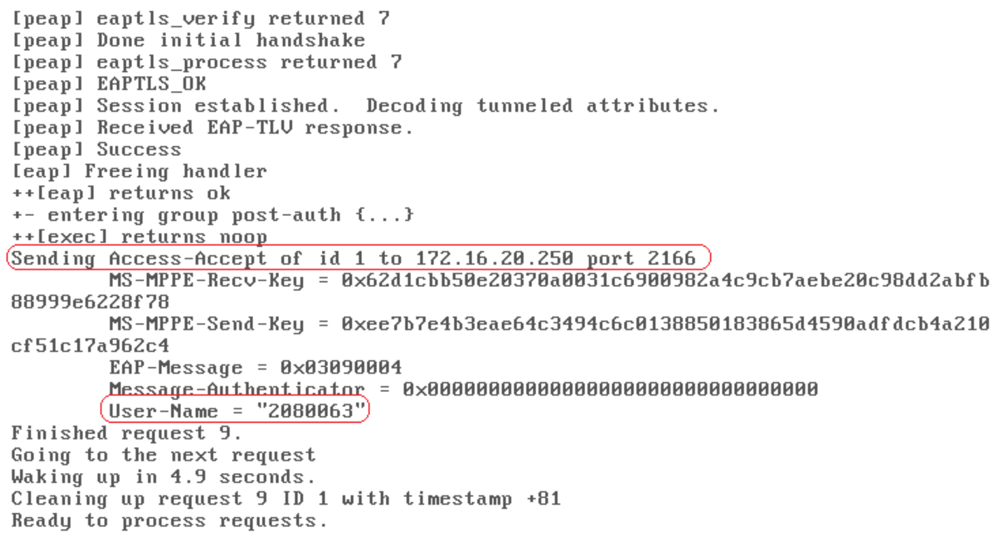
By using the tool ntlm_auth, it is possible to test some credentials validation with the active directory. If everything goes right a success message should be returned (Figure 41).



```
[root@VMCENTOS ~]# ntlm_auth --request-nt-key --domain=ipv6 --username=2080063
password:
NT_STATUS_OK: Success (0x0)
[root@VMCENTOS ~]# _
```

Figure 41 - ntlm_auth credential validation

By running FreeRADIUS in debug mode, it is possible to see and monitor (Figure 42) the authentication process status and validation by running the command '*radiusd –X*':

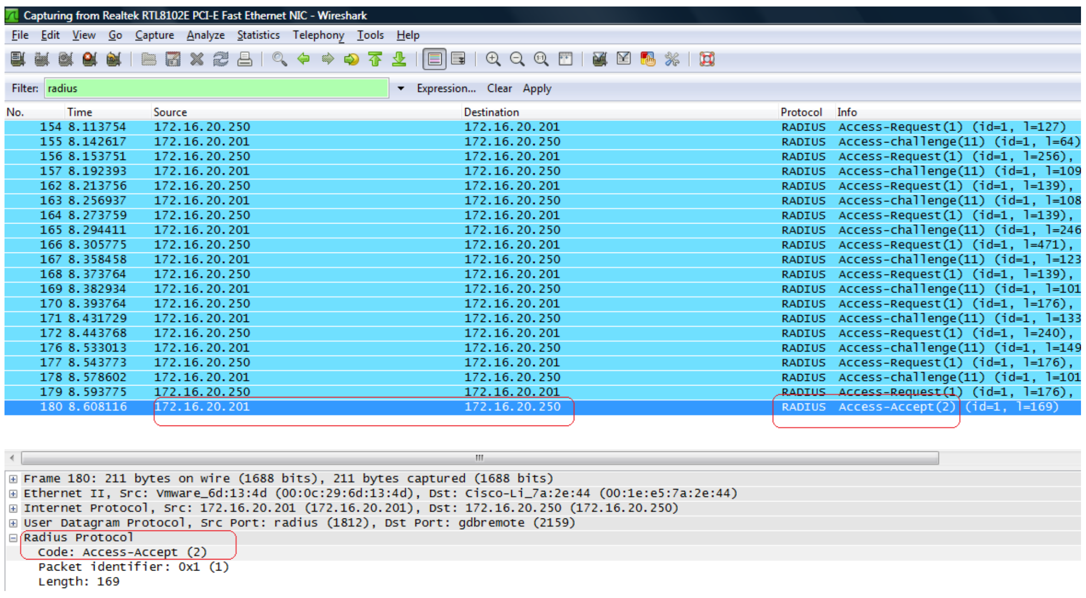


```
[peap] eaptls_verify returned 7
[peap] Done initial handshake
[peap] eaptls_process returned 7
[peap] EAPTLS_OK
[peap] Session established.  Decoding tunneled attributes.
[peap] Received EAP-TLV response.
[peap] Success
[eap] Freeing handler
++[eap] returns ok
+- entering group post-auth {...}
++[exec] returns noop
Sending Access-Accept of id 1 to 172.16.20.250 port 2166
        MS-MPPE-Recv-Key = 0x62d1cbb50e20370a0031c6900982a4c9cb7aebe20c98dd2abfb
88999e6228f78
        MS-MPPE-Send-Key = 0xee7b7e4b3eae64c3494c6c0138850183865d4590adfdcb4a210
cf51c17a962c4
        EAP-Message = 0x03090004
        Message-Authenticator = 0x00000000000000000000000000000000
        User-Name = "2080063"
Finished request 9.
Going to the next request
Waking up in 4.9 seconds.
Cleaning up request 9 ID 1 with timestamp +81
Ready to process requests.
```

Figure 42 - Output of RADIUS debug mode

Figure 43 shows the packets involved in an authentication process.

It is possible to see the RADIUS sequence packets (Access-Request – Challenge and Accept), as Figure 30 exhibits:


Capturing from Realtek RTL8102E PCI-E Fast Ethernet NIC - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply

Filter: radius

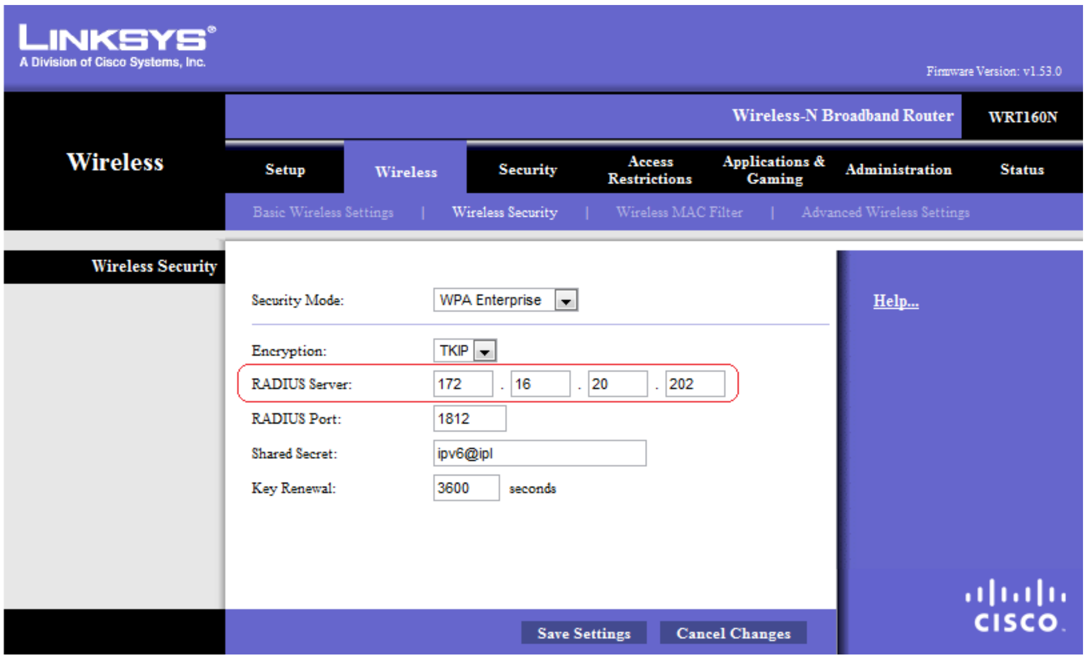
```
No.     Time        Source            Destination       Protocol Info
    154 8.113754    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=127)
    155 8.142617    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=64)
    156 8.153751    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=256),
    157 8.192393    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=109
    162 8.213756    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=139),
    163 8.256937    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=108
    164 8.273759    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=139),
    165 8.294411    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=246
    166 8.305775    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=471),
    167 8.358458    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=123
    168 8.373764    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=139),
    169 8.382934    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=101
    170 8.393764    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=176),
    171 8.431729    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=133
    172 8.443768    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=240),
    176 8.533013    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=149
    177 8.543773    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=176),
    178 8.578602    172.16.20.201     172.16.20.250     RADIUS   Access-challenge(11) (id=1, l=101
    179 8.593775    172.16.20.250     172.16.20.201     RADIUS   Access-Request(1) (id=1, l=176),
    180 8.608116    172.16.20.201     172.16.20.250     RADIUS   Access-Accept(2) (id=1, l=169)

⊞ Frame 180: 211 bytes on wire (1688 bits), 211 bytes captured (1688 bits)
⊞ Ethernet II, Src: Vmware_6d:13:4d (00:0c:29:6d:13:4d), Dst: Cisco-Li_7a:2e:44 (00:1e:e5:7a:2e:44)
⊞ Internet Protocol, Src: 172.16.20.201 (172.16.20.201), Dst: 172.16.20.250 (172.16.20.250)
⊞ User Datagram Protocol, Src Port: radius (1812), Dst Port: gdbremote (2159)
⊟ Radius Protocol
    Code: Access-Accept (2)
    Packet identifier: 0x1 (1)
    Length: 169
```

Figure 43 - RADIUS authentication

8.1.2 RADIUS in Windows Server 2008 – NPS

In this scenario, the access point (authenticator) will communicate directly with the Active Directory (IP=172.16.20.202) without using the FreeRADIUS server (Figure 44).

We used RADIUS in Windows Server 2008 by running the NPS service.

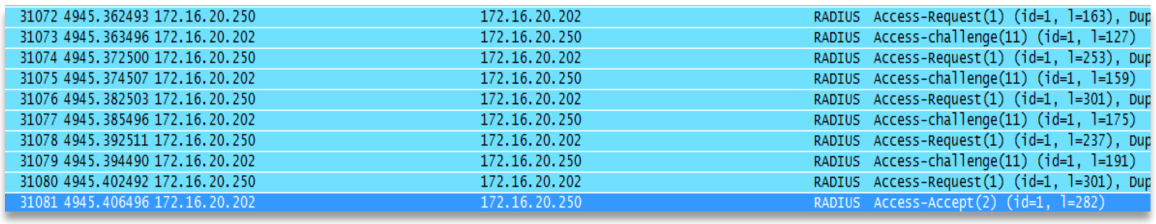

LINKSYS®
A Division of Cisco Systems, Inc.
Firmware Version: v1.53.0
Wireless-N Broadband Router
WRT160N
Wireless
Setup
Wireless
Security
Access Restrictions
Applications & Gaming
Administration
Status
Basic Wireless Settings | Wireless Security | Wireless MAC Filter | Advanced Wireless Settings
Wireless Security
Help...
Save Settings
Cancel Changes
cisco

Security Mode: WPA Enterprise
Encryption: TKIP
RADIUS Server: 172 . 16 . 20 . 202
RADIUS Port: 1812
Shared Secret: ipv6@ipl
Key Renewal: 3600 seconds

Figure 44 – AP Wireless Security configuration with NPS

By configuring the Network Policy Server (NPS) in Windows Server 2008, it is possible to enable the RADIUS feature, so that the server processes authentication requests.

In Figure 45, we can see a capture from a client access.



```
31072 4945.362493 172.16.20.250          172.16.20.202          RADIUS  Access-Request(1) (id=1, l=163), Dup
31073 4945.363496 172.16.20.202          172.16.20.250          RADIUS  Access-challenge(11) (id=1, l=127)
31074 4945.372500 172.16.20.250          172.16.20.202          RADIUS  Access-Request(1) (id=1, l=253), Dup
31075 4945.374507 172.16.20.202          172.16.20.250          RADIUS  Access-challenge(11) (id=1, l=159)
31076 4945.382503 172.16.20.250          172.16.20.202          RADIUS  Access-Request(1) (id=1, l=301), Dup
31077 4945.385496 172.16.20.202          172.16.20.250          RADIUS  Access-challenge(11) (id=1, l=175)
31078 4945.392511 172.16.20.250          172.16.20.202          RADIUS  Access-Request(1) (id=1, l=237), Dup
31079 4945.394490 172.16.20.202          172.16.20.250          RADIUS  Access-challenge(11) (id=1, l=191)
31080 4945.402492 172.16.20.250          172.16.20.202          RADIUS  Access-Request(1) (id=1, l=301), Dup
31081 4945.406496 172.16.20.202          172.16.20.250          RADIUS  Access-Accept(2) (id=1, l=282)
```

Figure 45 –NPS/RADIUS authentication

8.1.3 802.1x IPv6 compatibility

In order to adapt the first RADIUS scenario for using IPv6, the hardware equipment needs IPv6 compatibility.

Unfortunately our Cisco Access Points and Switches do not currently support RADIUS addressing over IPv6.

It is expected that the next IOS (Internetwork Operative System) Cisco release 12.2(58)SE will support RADIUS over IPv6. This new IOS should be published by Cisco in the next months.

Our theoretical IPv6 scenario address scheme should be as follows in Figure 46.

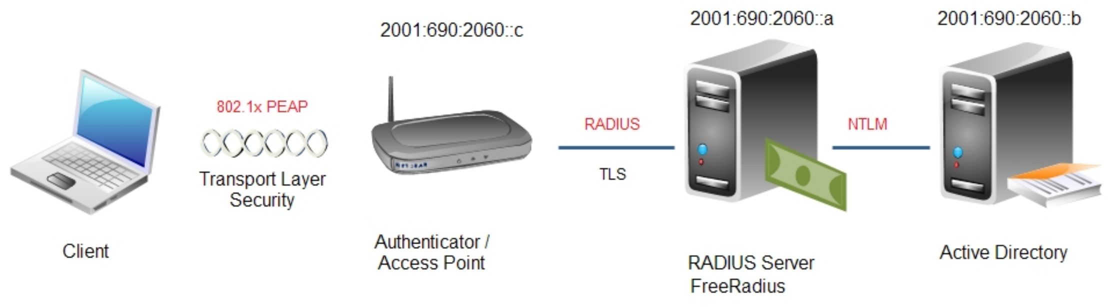
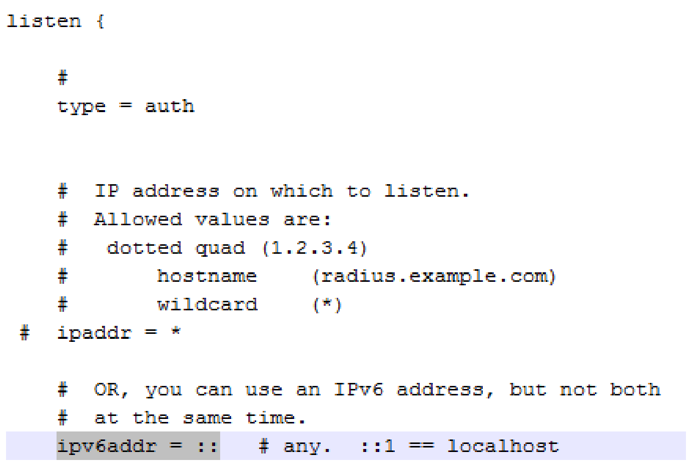


Figure 46 – RADIUS with AD – IPv6 address scheme

By default, FreeRADIUS creates a socket and listens for incoming request in IPv4.

In order to configure it for listening in IPv6, it is necessary to modify *radiusd.conf* (Figure 47) file by uncommenting the line “ *ipv6addr == ::* ”.

Obviously, FreeRADIUS should have a static IPv6 address defined manually (2001:690:2060::a).



```
listen {

    #
    type = auth


    #  IP address on which to listen.
    #  Allowed values are:
    #   dotted quad (1.2.3.4)
    #       hostname    (radius.example.com)
    #       wildcard    (*)
 #  ipaddr = *

    #  OR, you can use an IPv6 address, but not both
    #  at the same time.
    ipv6addr = ::    # any.  ::1 == localhost
```

Figure 47 - Configuration file radiusd.conf

In the clients.conf file, the client IP address (Authenticator/Access Point or Switch) must also be updated with its IPv6 address, as can be seen in Figure 48.

```
client 172.16.20.250 {
    secret    = ipv6@ipl
    shortname = 172.16.20.250
    nastype   = cisco
}

client 2001:690:2060::c {

    secret    = ipv6@ipl
    shortname = 2001:690:2060::c
    nastype   = cisco

}
```

Figure 48 - Configuration file clients.conf

Regarding the active directory, Windows Server also supports IPv6. So, it is necessary to give also a static IPv6 Address to its interface (2001:690:2060::b).

For the scenario with NPS in Windows Server 2008, using IPv6 should also work properly, as by default, NPS listens for RADIUS traffic on ports 1812, 1813, 1645, and 1646 for both IPv6 and IPv4 for all installed network adapters.

We implemented also following test scenario, which consists in testing the 802.1x IPv6 compatibility for wired access (port based), with a switch acting as Authenticator, as shown in the Figure 49:

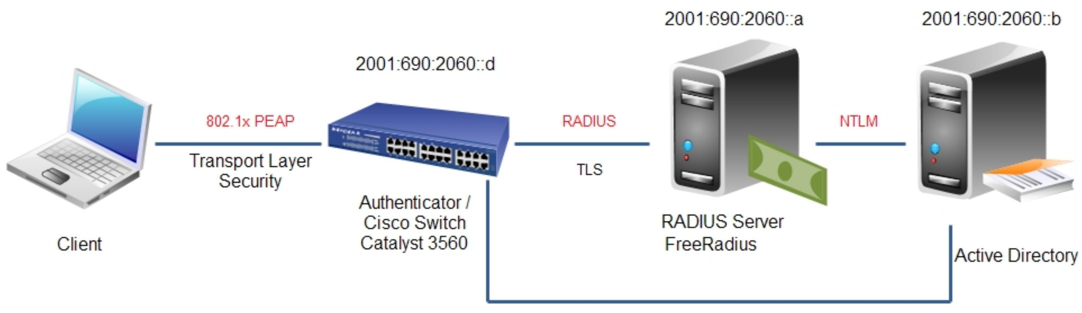


Figure 49 – RADIUS wired test scenario

In order to test our Switch IPv6 compatibility, we did an IOS upgrade to version 12.2 (55) SE3. This IOS supports IPv6 addressing commands. However, we noticed that it did not support RADIUS addressing over IPv6, as happened also with the Cisco Access Points.

Despite this lack of compatibility, we implement and list the proper Switch commands, which should run in the new IOS Cisco release, coming in the next months.

After entering the global configuration mode of the switch, here are the commands to activate IPv6 802.1x port based authentication:

- Activate AAA (Authentication, Authorization, Accounting)

```
# aaa new-model
```

- Create a list of authentication methods by using RADIUS group as default.

```
#aaa authentication dot1x default group radius
```

- Activate authorization for using dynamic VLAN (Virtual Lan) assignment by RADIUS.

```
#aaa authorization network default group radius
```

• Configure parameters of RADIUS server. In this case we use Radius IPv6 address 2001:690:2060::a and the default ports 1812 et 1813

Command not currently supported with IPv6 address

```
# radius server host 2001:690:2060::a auth-port 1812 acct-port 1813 timeout 3
```

• Configure the maximum number of retransmissions to the server for the requests (if there is no response of the server or if the server is slow).

```
#radius server retransmit 3
```

• Configure the shared secret between switch and RADIUS server.

```
#radius server key ipv6@ipl
```

•Now we must specify which interface port to operate in 802.1x mode. These following commands must be repeated for every port, which should do access control:

```
(config)#interface FastEthernet 0/20
(config-if)# switchport mode access
(config-if)# dot1x port-control auto
(config-if)# end
```

• Here are the IPv6 addressing interface commands

```
Switch(config)#interface Vlan1
Switch(config-if)#ipv6 address 2001:690:2060::d/64
Switch(config-if)#ipv6 nd ra suppress
```

The last command "*ipv6 nd ra suppress*" is needed to avoid the switch from doing Router Advertisements.

8.2 ICMPv6 Protocol protection

IPv6 brings in fact new features with ICMPv6 protocol, especially with the neighbor discovery protocol and router advertisement. However, this new feature can also be used for malicious attacks.

Let's see some ICMPv6 attack example and some solutions in order to protect the network from these threats.

8.2.1 Rogue RA messages

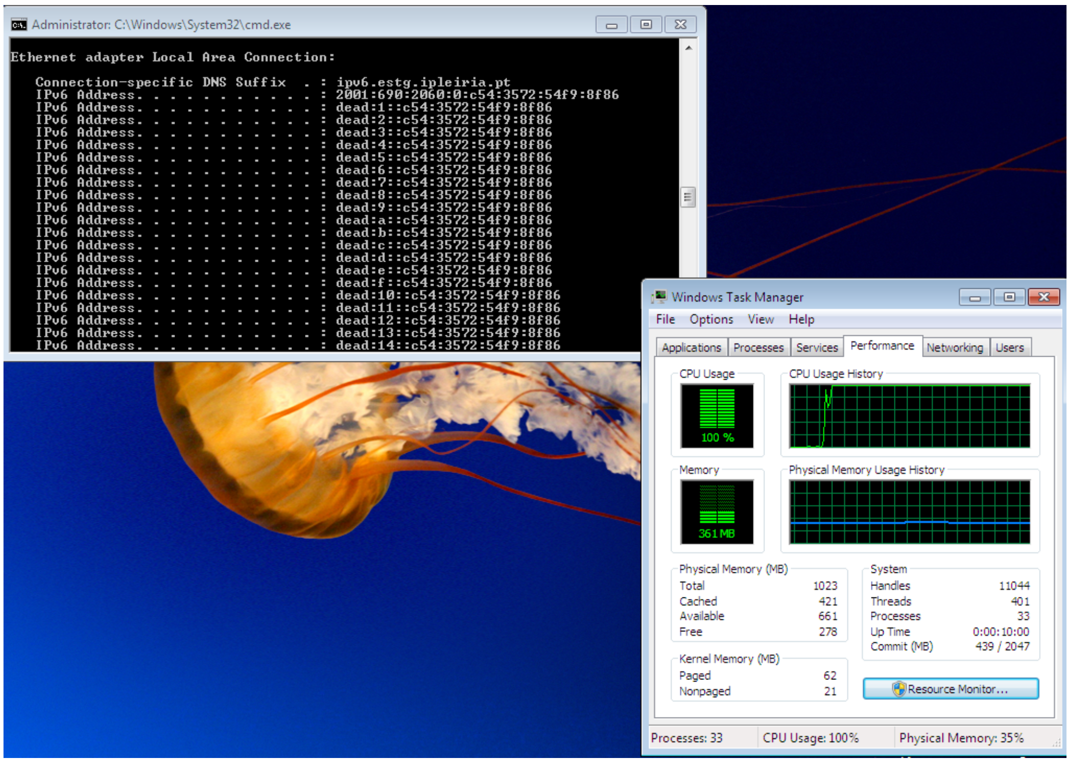
It is possible to cause a DOS (Denial of Service) attack in a network by flooding rogue RA messages.

In this test, we used a tool called npg, which allows to send and flood rogue RA messages [22].

This attack consists in announcing massive rogue RA messages, so that every device which receives these messages will auto-configure an IPv6 address, according the prefix announced. This cause in fact a significant increase of CPU usage, which can lead any target machine to crash!

Our first target machine was running Windows 7. As Figure 50 shows, the target victim has a lot of IPv6 address configured starting with the prefix ‘dead:’. The CPU usage of the victim rapidly rises to 100% during a long period.

We tested it also with other machines running Ubuntu 10.04 (Figure 51) and MacOS (Figure 52). Both systems were also infected. This attack affected all our LAN virtual machines. Also our Android Mobile phone was infected (Figure 53). So, this kind of attack can be a very dangerous threat for an entire real network!



```
Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : ipv6.estg.ipleiria.pt
   IPv6 Address. . . . . . . . . . . : 2001:690:2060:0:c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:1::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:2::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:3::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:4::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:5::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:6::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:7::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:8::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:9::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:a::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:b::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:c::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:d::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:e::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:f::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:10::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:11::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:12::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:13::c54:3572:54f9:8f86
   IPv6 Address. . . . . . . . . . . : dead:14::c54:3572:54f9:8f86
```

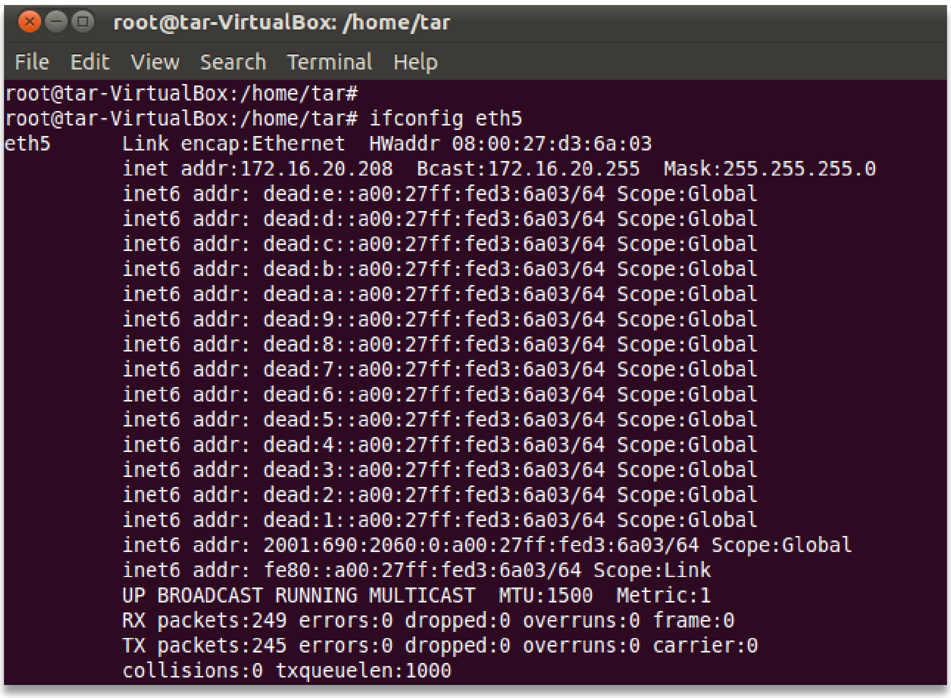
Physical Memory (MB)	
Total	1023
Cached	421
Available	661
Free	278

Kernel Memory (MB)	
Paged	62
Nonpaged	21

System	
Handles	11044
Threads	401
Processes	33
Up Time	0:00:10:00
Commit (MB)	439 / 2047

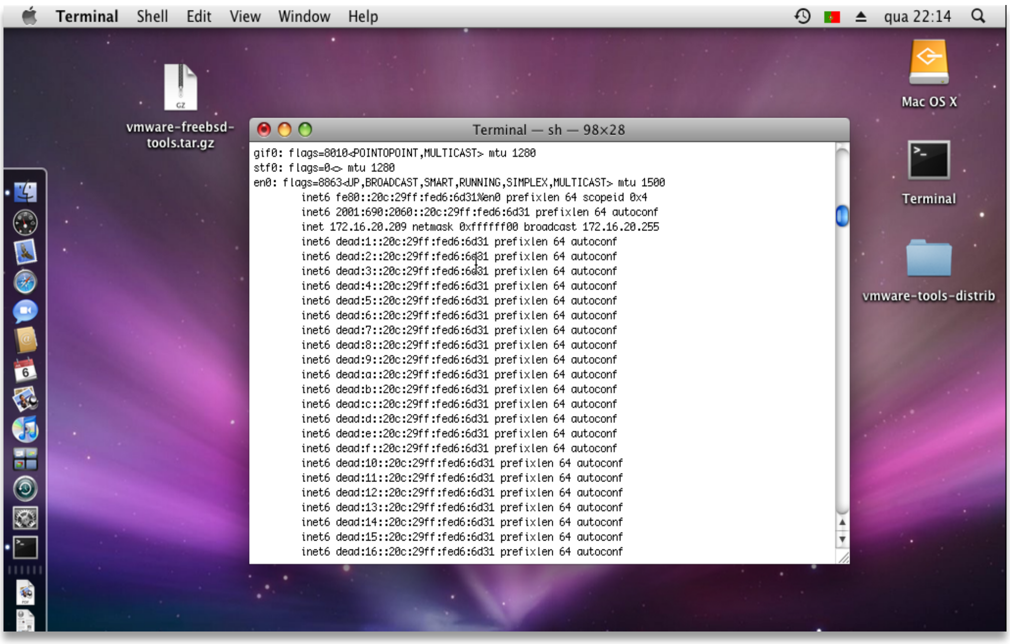
Processes: 33 CPU Usage: 100% Physical Memory: 35%

Figure 50 – Rogue RA vulnerability in Windows 7


root@tar-VirtualBox: /home/tar
File Edit View Search Terminal Help

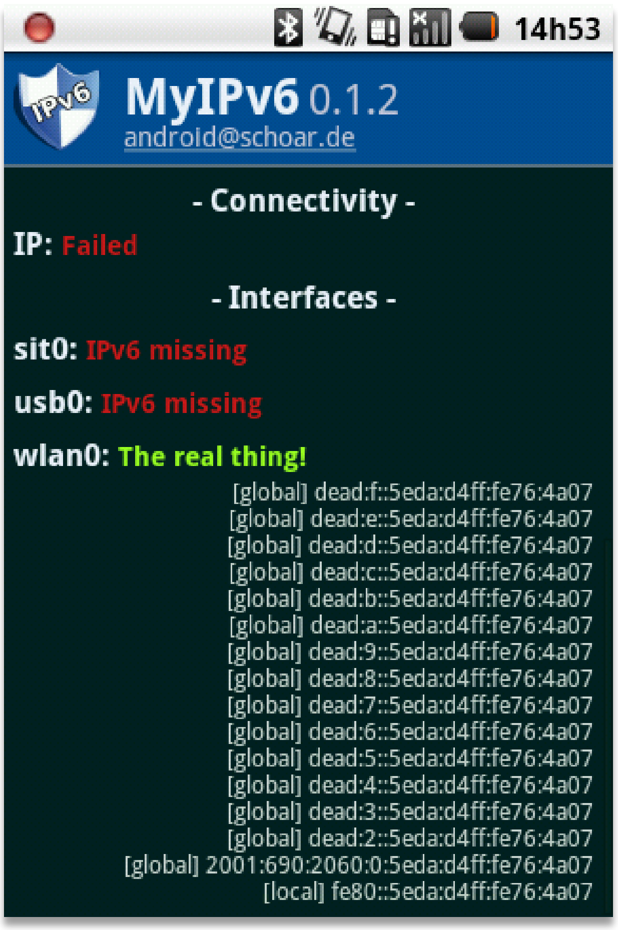
```
root@tar-VirtualBox:/home/tar#
root@tar-VirtualBox:/home/tar# ifconfig eth5
eth5      Link encap:Ethernet  HWaddr 08:00:27:d3:6a:03
          inet addr:172.16.20.208  Bcast:172.16.20.255  Mask:255.255.255.0
          inet6 addr: dead:e::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:d::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:c::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:b::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:a::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:9::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:8::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:7::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:6::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:5::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:4::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:3::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:2::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: dead:1::a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: 2001:690:2060:0:a00:27ff:fed3:6a03/64 Scope:Global
          inet6 addr: fe80::a00:27ff:fed3:6a03/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:249 errors:0 dropped:0 overruns:0 frame:0
          TX packets:245 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

Figure 51 – Rogue RA vulnerability in Ubuntu 10.04



```
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::20c:29ff:fed6:6d31%en0 prefixlen 64 scopeid 0x4
        inet6 2001:690:2060::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet 172.16.20.209 netmask 0xffffff00 broadcast 172.16.20.255
        inet6 dead:1::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:2::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:3::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:4::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:5::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:6::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:7::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:8::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:9::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:a::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:b::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:c::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:d::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:e::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:f::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:10::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:11::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:12::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:13::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:14::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:15::20c:29ff:fed6:6d31 prefixlen 64 autoconf
        inet6 dead:16::20c:29ff:fed6:6d31 prefixlen 64 autoconf
```

Figure 52 – Rogue RA vulnerability in Mac OS



14h53

MyIPv6 0.1.2
android@schoar.de

- Connectivity -

IP: Failed

- Interfaces -

sit0: IPv6 missing

usb0: IPv6 missing

wlan0: The real thing!

```
 [global] dead:f::5eda:d4ff:fe76:4a07
 [global] dead:e::5eda:d4ff:fe76:4a07
 [global] dead:d::5eda:d4ff:fe76:4a07
 [global] dead:c::5eda:d4ff:fe76:4a07
 [global] dead:b::5eda:d4ff:fe76:4a07
 [global] dead:a::5eda:d4ff:fe76:4a07
 [global] dead:9::5eda:d4ff:fe76:4a07
 [global] dead:8::5eda:d4ff:fe76:4a07
 [global] dead:7::5eda:d4ff:fe76:4a07
 [global] dead:6::5eda:d4ff:fe76:4a07
 [global] dead:5::5eda:d4ff:fe76:4a07
 [global] dead:4::5eda:d4ff:fe76:4a07
 [global] dead:3::5eda:d4ff:fe76:4a07
 [global] dead:2::5eda:d4ff:fe76:4a07
[global] 2001:690:2060:0:5eda:d4ff:fe76:4a07
       [local] fe80::5eda:d4ff:fe76:4a07
```

Figure 53 – Rogue RA vulnerability in Android

8.2.2 Security mechanisms

In order to prevent such attacks (like flooding Rogue RA messages), it is possible to configure the hardware to protect the network.

The RA guard feature for the Cisco hardware (IOS release 12.2(54)SG and 12.2(33)SXI4) allows to block any RA messages in specific ports by running the 'ipv6 nd-raguard' command for the related interface:

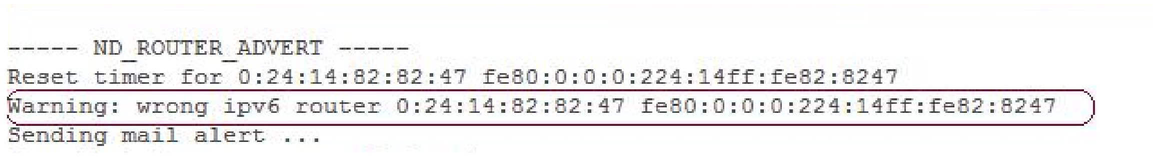
```
Router(config-if)# ipv6 nd raguard
```

There are other ways to restrict and protect the network. For example by configuring manually the address scope for RA messages in the firewall of Windows 7. But this solution seems not to be the proper one, as it is done in the client side.

Another solution would be also to reduce the target scope by using VLANs. This technique would consist in splitting the network infrastructure in into several VLANs with an IPv6 prefix for each of those VLANs. Each VLAN could contain for example a maximum of ten hosts. Thus an attacker could only attack nine other machines [23] .

We found also a tool called NDPMon, which analyzes all RA messages and checks their validity against an XML configuration file.

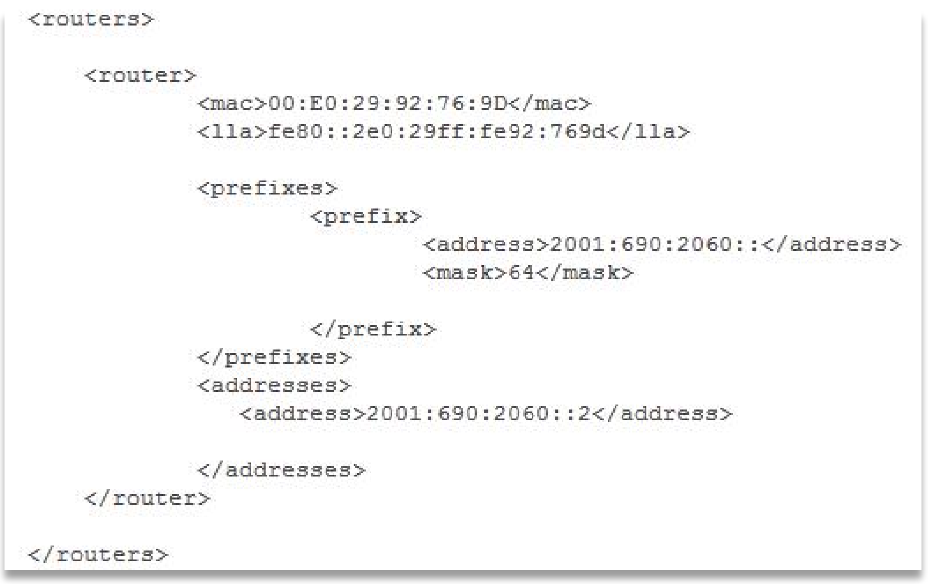
In Figure 54, we can see the NDPMon tool detecting an untrusted RA. In fact this RA comes from a router, which does not match our trusted server.



```
----- ND_ROUTER_ADVERT -----
Reset timer for 0:24:14:82:82:47 fe80:0:0:0:224:14ff:fe82:8247
Warning: wrong ipv6 router 0:24:14:82:82:47 fe80:0:0:0:224:14ff:fe82:8247
Sending mail alert ...
```

Figure 54 – NDPMon detecting untrusted RA

Figure 55 exhibits the NDPMon xml file, where we can see the datas (mac address, IP, prefix) of our trusted server.



```
<routers>

    <router>
            <mac>00:E0:29:92:76:9D</mac>
            <lla>fe80::2e0:29ff:fe92:769d</lla>

            <prefixes>
                    <prefix>
                            <address>2001:690:2060::</address>
                            <mask>64</mask>

                    </prefix>
            </prefixes>
            <addresses>
               <address>2001:690:2060::2</address>

            </addresses>
    </router>

</routers>
```

Figure 55 – NDPMon xml configuration file

8.3 Conclusions

Through the many scenarios described previously, we can conclude that the scenario of FreeRADIUS with AD could be a good solution for authenticating and restricting the user’s access in the network in a Dual Stack environment. Although the Cisco hardware (AP and Switches) are not currently compatible for addressing RADIUS server over IPv6, this should be resolved very soon with a new IOS release coming in the next months.

Regarding the security threat mentioned with RA flood attacks causing Denial of Service, the protection should be done carefully in the hardware equipments to restrict the RA messages. It is also very important to monitor the traffic in an IPv6 environment in order to detect such attacks.

Chapter 9

World IPv6 Day

On June 8th 2011, top websites and service providers such as Google, Yahoo, Facebook and Portugal Telecom joined together with thousands of others in a world-wide “test-flight” of IPv6.

For the first time, for a period of twenty four hours, a global trial was done, in which IPL participated. The goal was to motivate organizations across the industry (Internet service providers, hardware makers, operating system vendors and web companies) to prepare their services for IPv6 to ensure a successful transition as IPv4 address space runs out [24].

IPv6 has never been enabled at a global scale. World IPv6 Day helped industry players work together to support the new protocol on an accelerated timeline. With major web companies committing to enable IPv6 on their main websites, there are strong incentives for other industry players to ensure their systems are prepared for the transition [24].

One of the goals of World IPv6 Day was to expose potential issues under controlled conditions and address them as soon as possible. The vast majority of users should have been able to access services as usual, but in rare cases, misconfigured or misbehaving network equipment, particularly in home networks, could have impaired access to participating websites during the trial.

This chapter addresses preparations and results of that “test-flight”, both on an internal and on an external scope.

9.1 Preparation

In preparation for World IPv6 Day, it was asked of FCCN (the Internet Service Provider for IPL) to set up a native IPv6 connection to the Internet, as the previous connection had been disabled. A web server with DNS and a stateless auto-configuration solution was implemented, using an existing web-site. A secure Wi-Fi connection was used to allow access for students, teachers or anyone who wished to experience IPv6.

9.1.1 Equipment

Choice of equipment was due to availability. IOS versions were updated for each specific model to the latest available at the time (May 2011).

- **Switch**: Cisco Catalyst 3560, running IOS version 12.2(55)SE2
- **Access Point** : Cisco Aironet 1200, running IOS version 12.3(8)JEA3

9.1.2 Software

CentOS 5.6 (**C**ommunity **ENT**erprise **O**perating **S**ystem) is an Enterprise-class Linux Distribution derived from sources freely provided to the public by a prominent North American Enterprise Linux vendor. CentOS has numerous advantages over some of the other clone projects, including an active and growing user community, quickly rebuilt, and tested errata packages, an extensive mirror network, developers who are contactable and responsive, multiple free support avenues including IRC Chat, Mailing Lists, Forums, a dynamic FAQ [25].

CentOS is based on Red Hat Enterprise Linux (RHEL), and is widely used for server purposes throughout IPLeiria, and as such was an advised choice for implementing these services.

The following daemons were installed and configured:

- *Radvd* – Radvd and its stateless implementation was mentioned earlier.
- *Syslog-ng* – Multiplatform System Log daemon. Utilizing message parsing and classification, syslog-ng is able to correlate log messages both real-time and offline. It was used for logging user accesses through the wireless network.
- *Apache2* - Open-source IPv6-capable HTTP server for modern operating systems.
- *BIND* – The Internet Systems Consortium's (ISC) Berkeley Internet Name Domain daemon is an implementation of the DNS protocols and provides an openly redistributable reference implementation of the major components of the Domain Name System [26].
- *DHCPv4Server* – ISC's DHCPv4 server, used to auto-configure IPv4 addresses in our private network3.

For detailed configurations of the server and respective daemons, please refer to the Appendix.

9.1.3 Network Diagram

Figure 56 illustrates the network schematic, as well as the laboratory scenario that was set up.

[3] Refer to chapter 7 of this project, "Auto-configuration Tests"

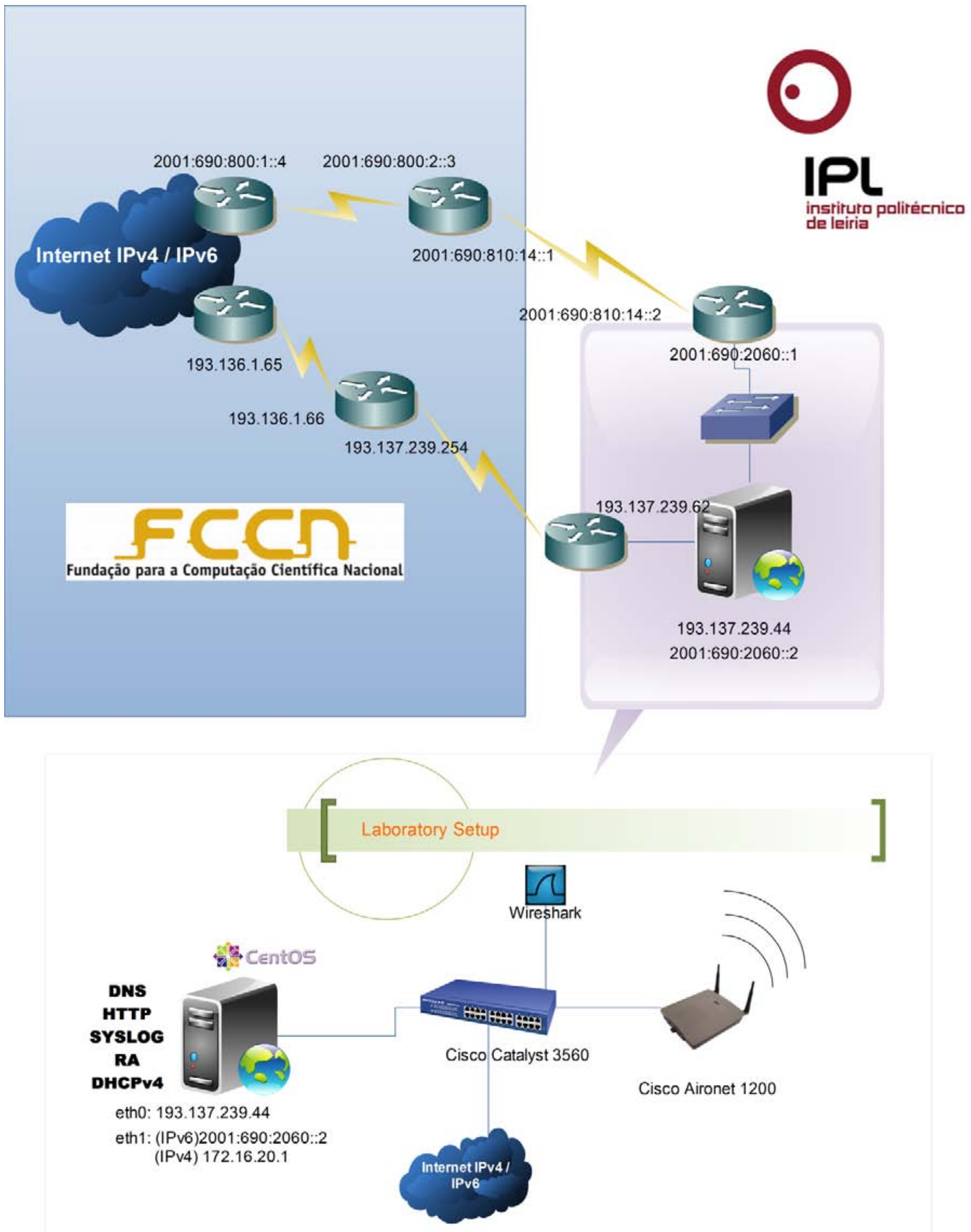


Figure 56 - IPLeiria Network Schematic

9.2 Local Results

As previously mentioned, IPLeiria contributed to the World IPv6 day effort by allowing access to a native wireless connection to the Internet for all interested. A *Syslog*[4] service was implemented in order to monitor user participation, and a counter was introduced on the website as to discriminate IPv4 and IPv6 traffic, which numbered 189 visitors over IPv6 and 1139 over IPv4 for that 24 hour period. No problems were reported.

9.3 Global Results

The best result possible was that the 8th of June 2011 passed without any major reported disturbances in the operation of the Internet. According to ISOC (Internet Society), the test demonstrated that "major websites around the world are well-positioned for the move to a global IPv6-enabled Internet, enabling its continued exponential growth", which also stated that "a key goal of World IPv6 Day was to expose potential issues with real-world IPv6 use under controlled conditions. Given the diversity of technology that powers the Internet, the global nature of the trial was crucial to identify unforeseen problems. The vast majority of users were able to access services as usual, but in rare cases, users experienced impaired access to participating websites during the trial [27].

Various measurements were taken by RIPE NCC. In Figure 57 we can see performance measurements performed on *Google*'s site.

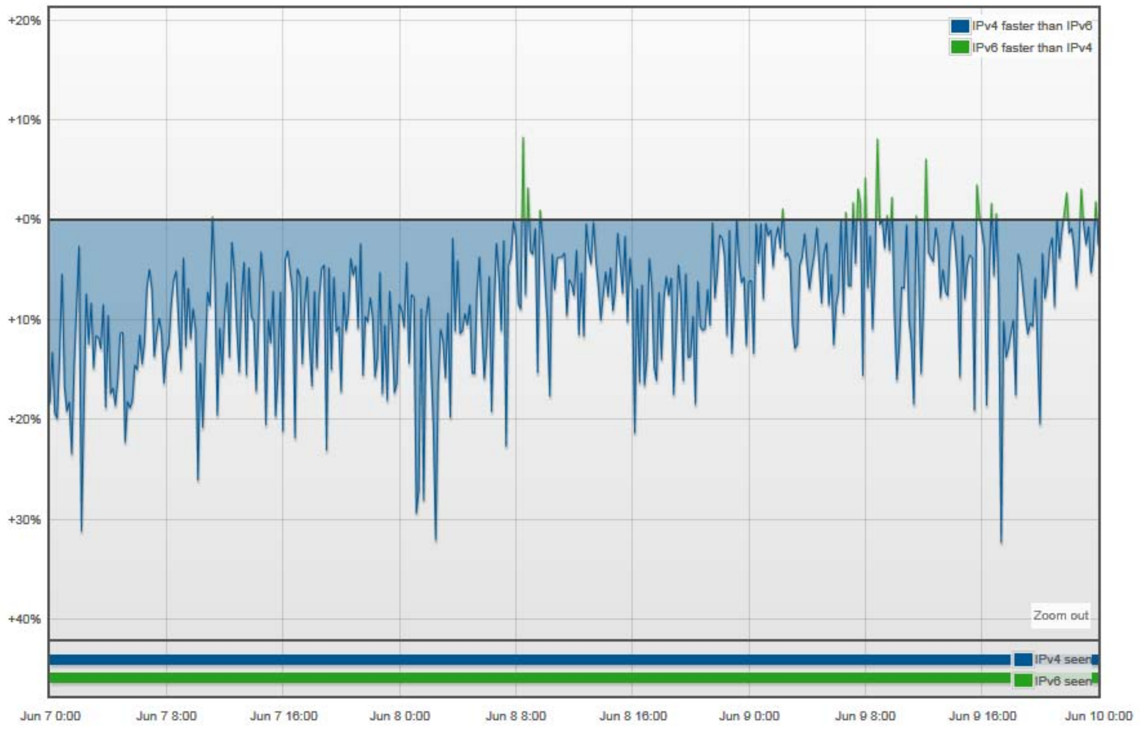


Figure 57 – IP performance as measured by RIPE NCC [28]

The graphic shows the relative performance between IPv6 and IPv4 pings. Values above zero mean that IPv6 performed better, values below zero mean that IPv4 performed better. Notice the spike on 8th June.

[4] Refer to the Appendix for logs and configurations.

Google took its own measurements, and, as can be seen in Figure 58, global IPv6 accesses are yet insignificant, totaling the small percentage of 0.32%.

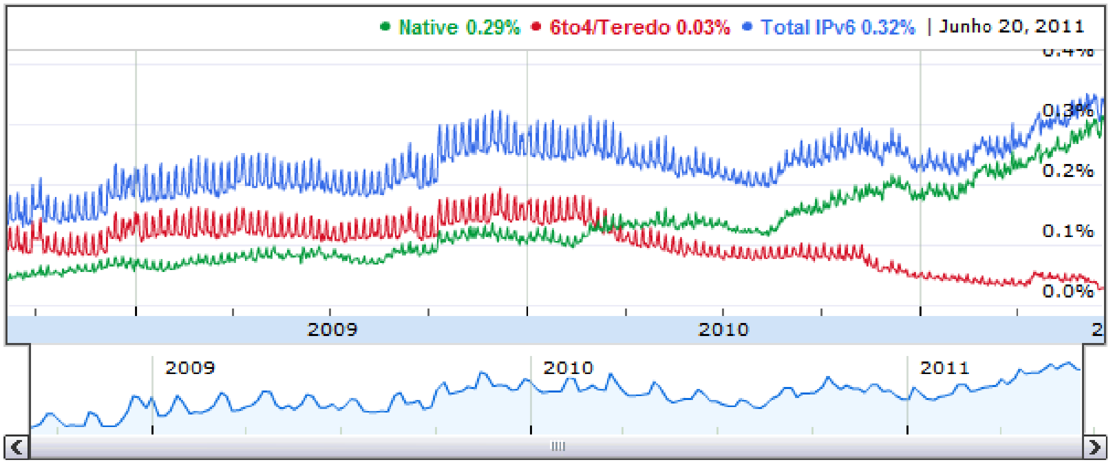


Figure 58 - Google Measurements of IPv6 traffic [29]

These results were also corroborated by *Arbor Networks*, who took their independent analysis of traffic (Figure 59) provided by six IPv6-capable service providers, during the 48 hours leading up to June 8th.

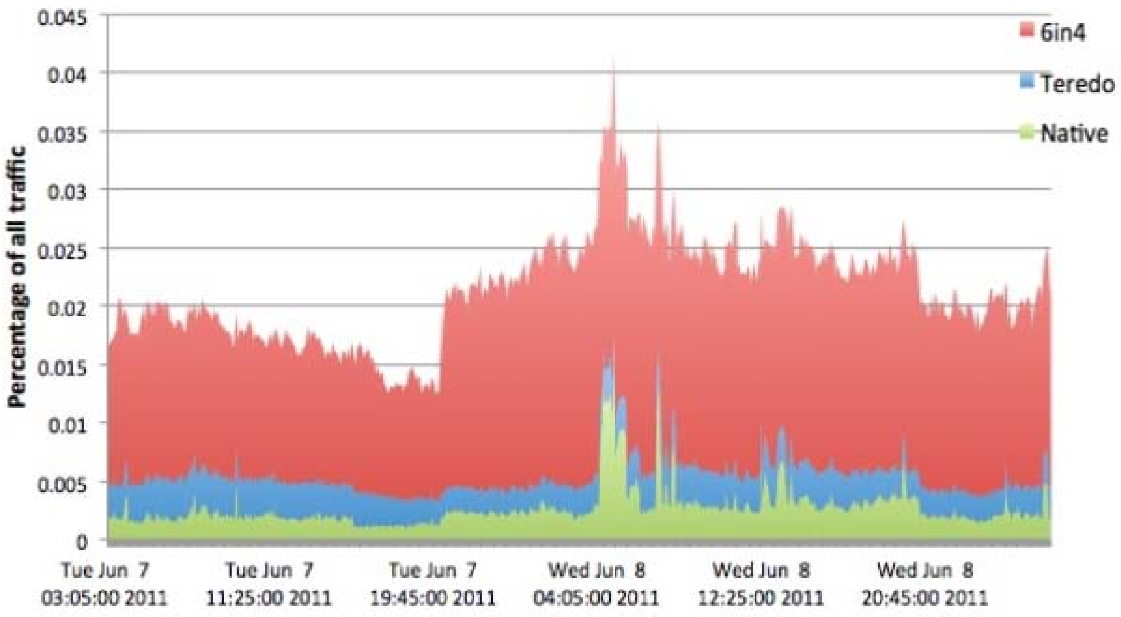


Figure 59 - Arbor Networks Measurement of IPv6 Traffic [46]

The following day *ComputerWorld* had such headlines as "*No news is good news on World IPv6 Day*", or "*Facebook: The Internet did not break*", and *eWeekEurope* reported "*IPv6 Day Is Hailed as a Qualified Success*".

Chapter 10

Conclusions

After studying and testing many scenarios related with the IPv6 client's configuration, we conclude that the proper solution at the moment for deploying an IPv6 wireless network should be a dual stack environment, involving both IPv6 and IPv4 protocols. Currently there are still some hardware and OS incompatibilities, which prevent from implementing some services only with IPv6. This problem has been encountered with providing DNS options to the client's machines, where in fact many OS do not support by default the information of nameserver given by DHCPv6 or with radvd. Moreover, android mobile phones need still to retrieve an IPv4 address to maintain an existing IPv6 connection.

Concerning the Authentication mechanism (RADIUS), similar problems were detected. In fact, the last current Cisco IOS (for Access Points and Switches) release does not support RADIUS addressing over IPv6. So we had to implement the RADIUS authentication mechanism over IPv4. This problem should be resolved in the next months, with a new Cisco IOS release, having RADIUS IPv6 addressing support. Our solution with dual stack grants however an efficient and secure authentication process.

For an organization with the scope of IPLeiria, not having to upgrade or replace already installed equipment seems like an advantage, and even if IPv6 does not provide all functions, these are, and will be for some time, transitional times. Access to an IPv4/IPv6 Internet is therefore assured at all times.

There are also some security risks and issues, which need to be monitored, like router advertisements for example. The auto-configuration process related with the ICMPv6 protocol can be seen as an advantage but it can also be a threat. In fact massive rogue RAs can be sent easily to lead the client machines to crash, if no proper security mechanisms are configured.

Regarding future work, we would suggest a study of the user's profiles authentication and management using VLANs. In fact, according the current academic context, many profiles could be implemented as for students, teachers, guest, etc.

Another interesting topic would be the Quality of Service (QoS) support in IPv6 wireless networks.

References

[1] Pete Loshin, *IPv6 Clearly Explained*.: Morgan Kaufmann, 1999.

[2] HighTeck.Net. [Online, Accessed April 2011]. http://www.highteck.net/EN/Network/Addressing_the_Network-IPv4.html

[3] Potaroo. [Online, Accessed April 2011]. http://www.potaroo.net/

[4] Tony Hain, "A Pragmatic Report on IPv4 Address Space Consumption," *Internet Protocol Journal*, Agosto 2005.

[5] Silvia Hagen, *IPv6 Essentials*.: O'Reilly, 2006.

[6] Cisco Systems, Inc. [Online, Accessed May 2011]. http://www.cisco.com

[7] Lisa Phifer, "The Trouble with NAT," *The Internet Protocol Journal*, December 2000.

[8] Internet World Stats. [Online, Accessed April 2011]. http://www.internetworldstats.com/stats3.htm

[9] Weigui Fang. Open Democracy. [Online, Accessed April 2011]. opendemocracy.net/people-china/article_1334.jsp

[10] Iljitsch van Beijnum, "Ipv6 internals," *The Internet Protocol Journal*, Setembro 2006.

[11] Mario Antunes and Luis Santos, Tópicos Avançados de Redes, 2010.

[12] S. Kawamura and M. Kawashima. (2010, August) RFC 5952 - A Recommendation for IPv6 Address Text Representation.

[13] R. Hinden and S. Deering. (2006, February) RFC 4291 - IP Version 6 Addressing Architecture.

[14] Global Unicast Address Format. [Online, Accessed June 2011]. http://www.tcpipguide.com/free/t_IPv6GlobalUnicastAddressFormat.htm

[15] Kukil Bora, "IPv6 What are the security issues it brings," *SiliconIndia News*, February 2011.

[16] Wikipedia. [Online, Accessed April 2011]. http://en.wikipedia.org/wiki/Neighbor_Discovery_Protocol

[17] S. Thomson and T. Narten. (1998, December) RFC 2462 - IPv6 Stateless Address Autoconfiguration.

[18] Sun Microsystems, System Administration Guide. [Online, Accessed July 2011]. http://dlc.sun.com/osol/docs/content/SYSADV3/ipv6-ref-34.html

[19] Cisco Systems, DHCP for IPv6. [Online, Accessed July 2011]. http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6554/ps6600/ps6641/aag_C45-456070_v2.pdf

[20] Director Hiroshi Esaki. WIDE Project. [Online, , Accessed July 2011]. http://www.wide.ad.jp/index.html

[21] S. Deering. (1991, September) RFC 1253 - ICMP Router Discovery Messages.

[22] NDISC6. [Online, Accessed July 2011]. http://www.remlab.net/ndisc6/

[23] T. Narten et al. (2007, September) RFC 4861.

[24] Project 11x: Rogue RA Attack with npg on Windows. [Online, Accessed June 2011]. http://samsclass.info/124/proj11/proj11x-RA-npg-win.html

[25] Eric Vyncke Scott Hogg, *IPv6 Security*.: Cisco Press, 2008.

[26] Internet Society. World IPv6 Day. [Online, , Accessed June 2011]. http://www.worldipv6day.org

[27] CentOS.org. [Online, , Accessed April 2011]. http://www.centos.org

[28] ISC. [Online, , Accessed April 2011]. http://www.bind9.net/

[29] Internet Society, "Successful World IPv6 Day Demonstrates Global Readiness for IPv6," *Internet Society Newsletter*, July 2011.

[30] RIPE NCC. [Online, Accessed June 2011]. http://v6day.ripe.net/cgi-bin/ping-rel.cgi?dst=2&dtype=d&region=x

[31] Google. Google IPv6 Statistics. [Online, Accessed June 2011]. http://www.google.com/intl/en/ipv6/statistics

[32] Cisco Systems, *IPv6 Extension Headers Review and Considerations (White Paper)*., 2006.

[33] Jeff Doyle, *CCIE Professional Development Routing TCP/IP*.: Cisco Press, 2006.

[34] Toni Soueid, "Présentation de Mobile IPv6," 2002.

[35] D Clark, L Chapin, V. Cerf, et al. (1991, December) RFC 1287 - Towards the Future Internet Architecture.

[36] R. Hinden and S Deering. (1995, December) RFC 1884 - IP Version 6 Addressing Architecture.

[37] R. Hinden and S. Deering. (1998, July) RFC 2373 - IP Version 6 Addressing Architecture.

[38] R. Hinden and Deering S. (1998, December) RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification.

[39] T. Narten, E. Nordmark, et al. (1998, December) RFC 2461 - Neighbor Discovery for IP Version 6 (IPv6).

[40] T. Narten and R. Draves. (2001, January) RFC 3041 - Privacy Extensions for Stateless Address Autoconfiguration in IPv6.

[41] R. Droms, J. Bound, et al. (2003, July) RFC 3315 - Dynamic Host Configuration Protocol for IPv6 (DHCPv6).

[42] R. Hinden and S. Deering. (2003, April) RFC 3513 - Internet Protocol Version 6 (IPv6) Addressing Architecture.

[43] O Troan and R. Droms. (2003, December) RFC 3633 - IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6.

[44] D Johnson, C. Perkins, et al. (2004, June) RFC 3775 - Mobility Support in IPv6.

[45] M. Cotton and L. Vegoda. (2010, January) RFC 5735 - Special Use IPv4 Addresses.

[46] Arbor Networks, [[Online, Accessed May

2011], http://asert.arbornetworks.com/2011/06/monitoring-world-ipv6-day/

Appendix

This appendix contains server configurations for the laboratory setup.

All these configurations were made on the server that was made available to us in the lab. The Operating System used was CentOs, version 5.6, and configurations are referenced here so that they can be of some future use.

1. Server Configurations

In order to boot straight to the command line, the default runlevel has to be set to 3. Runlevel 5 will boot to a GUI by default. In order to change this, the file "/etc/inittab" must be edited to contain:

```
id:3:initdefault:
```

Figure 60 - inittab

IPv6 has to be enabled. This is done by editing the file "/etc/modprobe.conf" to contain:

```
alias net-pf-10 on

alias ipv6 on
```

Figure 61 - modprobe.conf

and also in the file /etc/sysconfig/network, which also contains the ipv6 forwarding option, which needs to be enabled if a RA service is to run on this machine:

```
NETWORKING=yes
NETWORKING_IPV6=yes
FORWARD_IPV6=yes
IPV6FORWARDING=yes
HOSTNAME=ipv6.estg.ipleiria.pt
IPV6_DEFAULTGW=2001:690:2060::1
```

Notice IPv6 Forwarding

Figure 62 - Network Configuration

As for the network interfaces, the following configurations were made:

```
DEVICE=eth0
BOOTPROTO=none
BROADCAST=193.137.239.63
HWADDR=00:E0:29:47:21:79
IPADDR=193.137.239.44
IPV6INIT=yes
IPV6_AUTOCONF=yes
NETMASK=255.255.255.224
NETWORK=193.137.239.32
ONBOOT=yes
TYPE=Ethernet
GATEWAY=193.137.239.62
PEERDNS=yes
USERCTL=no
```

Figure 63 - Interface eth0 Configuration

```
DEVICE=eth1
BOOTPROTO=none
BROADCAST=172.16.20.255
HWADDR=00:E0:29:92:76:9D
IPADDR=172.16.20.1
IPV6ADDR=2001:690:2060::2/64
IPV6INIT=yes
IPV6_AUTOCONF=no
NETMASK=255.255.255.0
NETWORK=172.16.20.0
ONBOOT=yes
TYPE=Ethernet
```

Notice that both a global IPv6 address and a private IPv4 address are configured in this interface .

Figure 64 - Interface eth1 configuration

1.1 BIND9

Installing BIND:

```
$yum install system-config-bind
```

Configuring BIND:

```
$TTL 1H
@      SOA    ns     root.ns.ipv6.estg.ipleiria.pt. (  13
                                        3H
                                        1H
                                        1W
                                        1H )
@                                 IN    NS     ns.
                                  IN    A      193.137.239.44
                                  IN    AAAA   2001:690:2060::2
freeradius                        IN    AAAA   2001:690:2060::a
ipv6dc                                  IN       AAAA     2001:690:2060::b
ipv6dc                                  IN       A        172.16.20.202
freeradius                              IN       A        172.16.20.201
www                               IN    CNAME  @
ns                                IN    CNAME  @
```

Figure 65 - Zone File

```
$TTL 1H
@      SOA    ns.ipv6.estg.ipleiria.pt.  root.ns.ipv6.estg.ipleiria.pt.     (
       13
                                          3H
                                          1H
                                          1W
                                          1H )
       NS     ns.
44     PTR    ns.ipv6.estg.ipleiria.pt.
```

Figure 66 – Ipv4 DNS Reverse Zone File

```
$TTL 1H
0.0.0.0.0.6.0.2.0.9.6.0.1.0.0.2.ip6.arpa.      SOA
       ns.ipv6.estg.ipleiria.pt.  root.ipv6.estg.ipleiria.pt. (
                                          13
                                          3H
                                          1H
                                          1W
                                          1H )
       NS     ipv6.estg.ipleiria.pt.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   PTR    ipv6.estg.ipleiria.pt.
a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 PTR      freeradius.ipv6.estg.ipleiria.pt.
b.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 PTR      ipv6dc.ipv6.estg.ipleiria.pt.
```

Figure 67 - IPv6 DNS Reverse Zone File

```
options {
       listen-on-v6 { any; };
       forward first;
forwarders {
       2001:690:a00:4001::200;
       2001:690:a80:4001::200;
       2001:690:a00:4001::100;
       193.137.239.226;

};
directory "/var/named";
       dump-file "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
       /*
        * If there is a firewall between you and nameservers you want
        * to talk to, you might need to uncomment the query-source
        * directive below.  Previous versions of BIND always asked
        * questions using port 53, but BIND 8.1 uses an unprivileged
        * port by default.
        */
        // query-source address * port 53;
};
controls {
       inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};
zone "0.0.0.0.0.6.0.2.0.9.6.0.1.0.0.2.IP6.ARPA." IN {
       type master;
       file "2001:690:2060:.db";
};
zone "239.137.193.IN-ADDR.ARPA." IN {
```

Option to listen to IPv6 requests is required

External DNS servers, both for IPv6 and IPv4

```
       type master;
       file "193.137.239.db";
};
zone "ipv6.estg.ipleiria.pt." IN {
       type master;
       file "ipv6.estg.ipleiria.pt.db";
};
zone "." IN {
       type hint;
       file "named.root";
};
zone "localdomain." IN {
       type master;
       file "localdomain.zone";
       allow-update { none; };
};
zone "localhost." IN {
       type master;
       file "localhost.zone";
       allow-update { none; };
};
zone "0.0.127.in-addr.arpa." IN {
       type master;
       file "named.local";
       allow-update { none; };
};
zone
"0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa."
IN {
        type master;
       file "named.ip6.local";
       allow-update { none; };
};
zone "255.in-addr.arpa." IN {
       type master;
       file "named.broadcast";
       allow-update { none; };
};
zone "0.in-addr.arpa." IN {
       type master;
       file "named.zero";
       allow-update { none; };
};
include "/etc/rndc.key";
```

Figure 68 - named.conf

1.2 Apache

Installing Apache:

```
$yum install dhcpv6
```

Configuring Apache:

The httpd.conf file must contain the following entry, in order to respond to HTTP requests over IPv6:

```
# Listen: Allows you to bind Apache to specific IP addresses
and/or
# ports, in addition to the default. See also the
<VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown
below to
# prevent Apache from glomming onto all bound IP addresses
(0.0.0.0)
#
#Listen 12.34.56.78:80
Listen *:80
```

Option to listen to requests on *ANY* kind of address is required

Figure 69 - httpd.conf

1.3 RADVD

Installing radvd:

```
$yum install radvd
```

Configuring radvd:

```
# NOTE: there is no such thing as a working "by-default" configuration
file.
#       At least the prefix needs to be specified.  Please consult the
radvd.conf(5)
#         man page and/or /usr/share/doc/radvd-*/radvd.conf.example for
help.
#
#
interface eth1
{
        AdvSendAdvert on;
         AdvManagedFlag off;
         AdvOtherConfigFlag on;
        MaxRtrAdvInterval 80;
        MinRtrAdvInterval 60;
        prefix 2001:690:2060::/64
        {
                AdvOnLink on;
                AdvAutonomous on;
        };

#        RDNSS 2001:690:2060::2
#       {
#           AdvRDNSSPreference 8;
#           AdvRDNSSOpen off;
#
#       };
};
```

Commented lines could be used for pure stateless auto-configuration (where possible)

Figure 70 - Radvd.conf

1.4 DHCPv4

Installing DHCP:

```
yum install dhcp
```

Configuring DHCP:

```
ddns-update-style none;
# option definitions common to all supported networks...
option domain-name "ipv6.estg.ipleiria.pt";
option domain-name-servers 172.16.20.1;
default-lease-time 600;
max-lease-time 7200;
subnet 172.16.20.0 netmask 255.255.255.0 {
      range 172.16.20.1 172.16.20.254;
      option routers 172.16.20.1;
}
```

Internal IPv4 addressing was used for DNS queries on the private network to function upon.

Figure 71 - dhcpd.conf

1.5 DHCPv6

Installing dhcpv6:

```
$yum install dhcpv6
```

Configuring dhcvp6:

```
interface eth1 {
    server-preference 255;
    renew-time 60;
    rebind-time 90;
    prefer-life-time 130;
    valid-life-time 200;
    allow rapid-commit;
    option dns_servers 2001:690:2060::2;
 #   link AAA {
 #       pool{
 #           range 2001:690:2060::3/64 to 2001:690:2060::10/64;
 #
 #       };
        #range 3ffe:501:ffff:1::1 to 3ffe:501:ffff:1::10/64;
#    };
```

Commented lines could be used for Stateful auto-configuration (where possible)

Figure 72 - dhcp6s.conf

1.6 Syslog-ng

To install Syslog-ng, the repository needs to be added:

su -c 'rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm'

Then the daemon can be installed:

```
$yum install syslog_ng
```

File /etc/syslog-ng/syslog-ng.conf was edited as such:

```
source s_net { udp(ip(0.0.0.0) port(514)); }; #0.0.0.0 will
bind to all interfaces on your syslog server.
destination d_cisco { file("/var/log/cisco.log"); };
log { source(s_net); destination(d_cisco); };
```