Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea Magistrale in Informatica

Migration strategies from IPv4 to IPv6 in a complex service provider environment

Tesi di Laurea in Reti di Calcolatori

Relatore:
Gabriele D'Angelo

Presentata da:
Luca Toscano

Co-Relatore:
Ing. Denis Pavani

II Sessione
Anno Accademico 2011-2012

"So long, and thanks for all the fish!"

The hitchhiker's guide to the galaxy

Sommario

Il protocollo di rete IPv6 è una delle tecnologie più sottovalutate e rimandate della storia informatica. L'abbondanza di indirizzi IPv4 ha sempre relegato la migrazione ad IPv6 nell'angolo delle cose da fare, ma il 2012 potrebbe rappresentare una svolta. L'Autorità che regolamenta l'assegnazione degli indirizzi IP (IANA) ha terminato nel 2011 il pool di indirizzi IPv4 disponibili, e quest'anno (2012) la stessa sorte è toccata ai Registri Internet Regionali (RIR) RIPE e APNIC. Come se ciò non bastasse l'aumento vertiginoso degli acquisti di tablet e cellulari di ultima generazione pone una ulteriore nota di gravità, in quanto presto la domanda di connettività Internet supererà la disponibilità di indirizzi IPv4. Soluzioni come Network Address Translation saranno utili in un primo step per gestire la situazione, ma la soluzione non è ovviamente scalabile nel lungo periodo. Giganti come Google e Facebook si sono portati avanti e sono diventati i punti di riferimento di questa rivoluzione tecnologica, ma il mercato globale è ancora restio a investire le risorse necessarie per favorire l'adozione di IPv6.

Il CINECA è un consorzio interuniversitario italiano molto famoso, rappresenta il ponte tecnologico tra università italiana, ricerca, industria e pubblica amministrazione: la sua posizione richiede molta sensibilità nell'anticipare l'avvento di nuove tecnologie, motivo per cui il Dipartimento di Servizi e Tecnologie (DSET) ha deciso di iniziare la migrazione verso IPv6 il prima possibile. Ho affiancato per alcuni mesi sistemisti, tecnici specializzati in reti informatiche e programmatori del CINECA nel tentativo di stabilire una strategia coerente con i bisogni dell'azienda.

Questa tesi è il risultato di tutto il lavoro svolto: scelte tecniche, problemi incontrati e risultati ottenuti. Nella prima parte vengono illustrati il protocollo IPv6 e l'infrastruttura del CINECA, mentre nella seconda viene descritto il processo che ha portato alla creazione della strategia di migrazione da IPv4 ad IPv6. Come si può facilmente immaginare l'obiettivo non era quello effettuare tutto il lavoro richiesto dalla migrazione, piuttosto quello di creare fondamenta solide su cui appoggiarsi nei mesi di lavoro successivi per arrivare all'adozione completa del protocollo IPv6.

Introduction

For more than a decade the IPv6 protocol has represented only a wonderful unnecessary technology, because of the pervasive presence of IPv4 and the abundance of its addresses guaranteed by the CIDR subnetting policy. By the time I am writing this thesis, August 2012, the Internet Assigned Numbers Authority (IANA) has depleted its IPv4 address pool and all the Regional Internet Registries like RIPE and APNIC have left only tens of millions of spare addresses. Moreover smartphones, pervasive Internet social networks and cloud services like the Google Apps have changed the game, in the near future everyone will require an IP address to be always *online*. As the researcher Geoff Huston said:

> *The fancy part of Internet needs more addresses!*[1]

It is obvious that the IPv4 stack will not be deprecated in years but it is now time to start using IPv6 in order to acquire the necessary knowlegde to facilitate the migration. But what does it mean migrating to IPv6? Does it concern only the network infrastructure or not? How should the Internet services and applications change in order to embrace properly the new protocol? I worked several months for the CINECA Interuniversity Consortium to answer those questions, and the results are described in this thesis.

CINECA is an Italian no profit consortium consisting of 54 Italian universities, the Istituto Nazionale di Oceanografia e Geofisica sperimentale (OGS), the Italian National Research Council (CNR) and the Italian Ministry for Education, University and Research (MIUR); it is an high technology bridge

[1] Geoff Huston, RIPE meeting, Rome, 15-19 November 2011

among the academic world, the research, the industry and the public administration.

In the first part I will present briefly the IPv6 protocol and the CINECA infrastructure in order to give more information to the reader about the environment I worked into. In the second part of thesis I will explain all the work I have done together with the network specialists and software engineers working at CINECA to come up with a strategy for the migration to IPv6.

Contents

List of Figures

List of Tables

Part I

IPv6 and CINECA

Chapter 1

Introduction to IPv6

Any sufficiently advanced technology is indistinguishable from magic.
Sir Arthur C. Clarke

1.1 A brief history of the Internet Protocol

The first pioneering paper [9] on the Internet Protocol was published in 1974, after a decade it became the Institute of Electrical and Electronics Engineers[1] (IEEE) RFC 791 [1], namely the actual IPv4, running the Internet since then. The IPv4 was first recognized as a bottleneck during 1992 because of the rapid worldwide Internet adoption and expansion, its address range was not designed for such an enormous scalability. The first patch to the problem was the introduction of the Classless Inter Domain Routing (CIDR) [14] by the Internet Engineering Task Force[2] (IETF) in 1993, its aim was to replace the inefficient classful addressing architecture of IPv4 in order to slow down the rapid exhaustion of its addresses. Each IPv4 address is composed by two groups of bits: the network address, which identifies a whole subnet, and the host address, which identifies a specific interface on a host connected to that subnet. The classful approach permits to use a network address composed by groups of 8 bits only, whereas the CIDR one permits a variable length number

[1] `http://www.ieee.org`
[2] `http://www.ietf.org`

of bits, a more flexible solution to design efficient network addressing plans. The CIDR RFC was published in 2006, it has slowed down the IPv4 address exhaustion but since the beginning the IETF knew it would have been only a temporary solution to the problem. This is why the IETF began to work on a new Internet protocol during 1992 and it published the definition of the IPv6 [11] protocol in December 1998.

1.2 The IPv4 address exhaustion

By the time this thesis has been written[3] the Internet Assigned Numbers Authority [4] (IANA) has finished all of its IPv4 address ranges available and the statuses of the Regional Internet Registries (RIRs) is depicted in Table 1.1. This low availability of IPv4 addresses should be considered together

RIR name	**IPv4 addresses available** (Millions)
RIPE	28
APNIC	18
AfriNIC	56
ARIN	104
LACNIC	48

Table 1.1: IPv4 addresses available in late 2012 by all the RIRs

with the recent rise of the smartphone and tablet markets, because of those kind of devices are meant to be always connected, so they need addresses. This coincidence of events should bring Internet Service Providers (ISPs) and the Internet content delivery business to adopt IPv6 in the near future in order to keep customers satisfied. The risk is ISPs will use old fashioned solutions like the Network Addressing Translation (NAT) to slow down the

[3] August 2012

[4] `http://www.iana.org`

IPv6 transition, but surely this will be only a temporary fix and not a stable solution.

1.3 The actual adoption

Despite the lack of address availability from the RIRs the IPv6 protocol is used[5] only by a small percentage of the Internet, but all the statistics show an impressive spike for the year 2012, as we can see in Figure 1.1 and Figure 1.2. The increment of IPv6 connections is related to many factors, like the

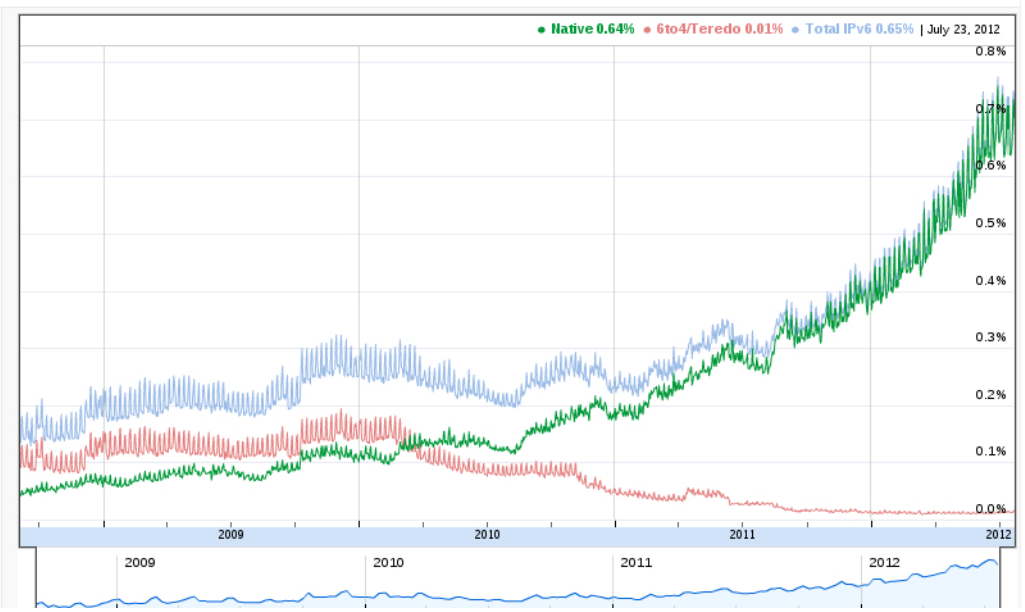

Figure 1.1: IPv6 traffic measured by Google in July 2012. The green line indicates native IPv6 connections.

IPv4 address exhaustion stated in Section 1.2, the mature IPv6 support from network vendors like CISCO and the recent World IPv6 Day[6] supported by Web giants like Google and Facebook.

[5] late 2012

[6] `http://www.worldipv6launch.org`

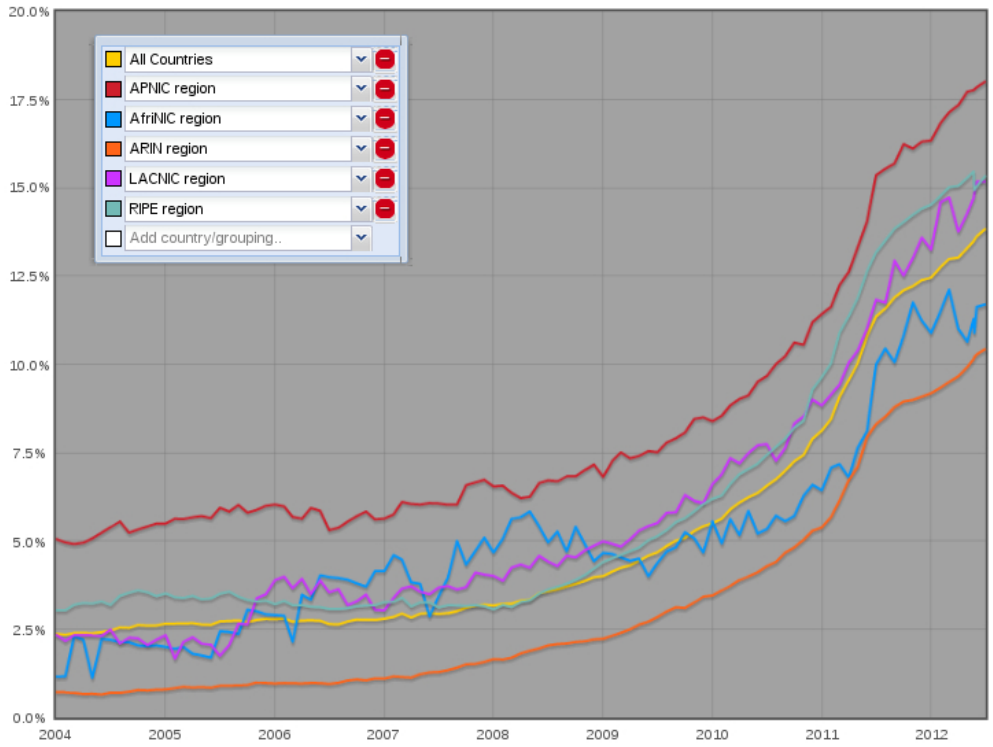

Figure 1.2: IPv6 traffic measured by the RIPE RIR (July 2012)

1.4 Do I need IPv6?

The title of this section is one of the most recurrent networking question a company will ask to its IT team. The answer should be a straight *Yes!* but for the sake of clarity it is better to make some distinctions. The IPv4 address exhaustion concerns only public addresses, therefore an internal network is not going to run out of connectivity. Conversely the Internet services exposed by a company are going to receive IPv6 traffic during the next years because the RIRs will soon run out of IPv4 public connectivity (this scenario does not take into account awful NAT tricks by ISPs) and therefore the same will happen to ISPs. There will be neither a day for the IPv4/IPv6 switch nor one for an IPv4 shutdown, instead there will be a gradual migration to a dual stack solution and hopefully IPv4 will be dismissed definitely some day not too far. The key point is to actively embrace the IPv6 technology, investing

time and resources to gain knowledge and expertise in the field, instead of passively wait until it will be mandatory.

1.5 The protocol

The IPv6 protocol is not only a tool to provide more addresses, it is also an efficient redesign of IPv4, namely many improvements and desired features collected through the years by network engineers and architects. This section is not meant to be a complete technical presentation of IPv6, but a short list of key features that should give a quick summary of the importance of this new protocol (the interested reader is invited to consult the related RFC [11]).

1.5.1 A new addressing scheme

The first big change in IPv6 is the address length, that is four times longer than IPv4: 128 bits. The total number of available addresses is enormous, it will be possible to map thousands devices for each squared meter of Earth. This feature will definitely boost up the rise of the Internet of Things, namely the possibility to give an IP address to each electronic device equipped with sensors, even the things we use in everyday life: refrigerators, dishwashers, ovens, medical equipment, mp3 players and so on.

The IPv6 address is a lot different than IPv4 in several aspects:

- each group of eight bits is represented in hexadecimal format;
- each group of sixteen bits, called a nibble, is separated by a colon;
- the leading zeros of a nibble could be omitted;
- multiple all zeros nibbles could be represented by two colons (::).

Here are some examples:

- **complete address** 2001:0000:0000:0001:0250:56ff:fe9a:72d5

- **leading zeros omitted** 2001:0:0:1:250:56ff:fe9a:72d5
- **two nibbles shortened** 2001::1:250:56ff:fe9a:72d5

The wary reader should follow this simple addressing exercise to convince himself about the enormous IPv6 address space:

- a company has received from the RIPE RIR a /48 subnet prefix for its network;
- following the best practice (see Section 1.5.5) it has reserved the last 64 bits to identify each host of its network;
- the company has now the possibility to use 16 bits to address its subnets, that is two to the power of 16 subnets, 65536;
- moreover each subnet could address two to the power of 64 hosts, that is 18446744073709551616 hosts.

The reader is invited also to notice that in a /32 IPv6 address prefix the number of available subnets is equal to the actual public Internet address space.

1.5.2 One interface, multiple addresses

The original IPv4 stack had one constant: one physical interface manages only one IPv4 address. In the past years many operating systems like Linux and Windows implemented the so called *IP aliasing*, a technique able to handle multiple IPv4 addresses on the same interface emulating it through software. In the IPv6 world an interface could hold multiple addresses, each one having its scope in the network, implementing the IP aliasing concept directly in the protocol specifications.

There are two big families of IPv6 addresses, the unicast one and the multicast one; in this paragraph we will analyze the former, whereas in Section 1.5.3 the latter.

Type	Subnet prefix	Scope
Link Local	fe80::/64	link
Unique Local	fd00::/8	cooperating sites
Global	not fixed	Internet

Table 1.2: Unicast IPv6 address types

As stated in Table 1.2 there are three types of unicast address: Link Local, Unique Local and Global. The first one is the entry point of the host to a network, because it is the address used to communicate through the link directly connected to the physical interface (it is auto generated using a reserved subnet prefix and the host identifier, either IEEE EUI-64 one or the random one, see Section 1.5.5). The Unique Local address is the IPv6 counterpart of the IPv4 private address[7] whereas the Global address is an Internet routable address. The reader is invited to notice how flexible is this solution, because it permits an interface to hold more than one address and to use them depending on its communication scope.

Type	Example
Link Local	fe80::215:58ff:fe83:9f08/64
Unique Local	fd00:1A86::215:58ff:fe83:9f08/64
Global	2001:760:2e0a:0:215:58ff:fe83:9f08/64

Table 1.3: An example of three different IPv6 addresses for the same interface

Table 1.3 shows some examples of unicast addresses, the reader is invited to compare them with Table 1.2.

1.5.3 IPv6 multicast

A multicast address is an identifier related to a group of interfaces on hosts willing to receive the same set of IP packets. An host has to subscribe

[7]IPv4 Class A private address range 10.0.0.0/8, IPv4 Class B private address range 192.168.0.0/16

one of its interfaces to a multicast group in order to receive and send packets to a specific set of hosts, a more flexible solution compared to broadcasts. Multicast is an optional feature in IPv4, added through the years in order to implement some services efficiently. In IPv6 they are a first class passenger, it is used for all the vital network operations instead of broadcast.

Bits position	8	4	4	112
Address Bits	FF	00PT	scope	group ID

Table 1.4: The IPv6 multicast address format

A multicast address follows the format stated in Table 1.4:

- the first eight bits are the fixed subnet prefix, namely **ff00::/8**;
- the second group of bits has two flags, P and T, respectively Prefix and Temporary. They indicate whether or not the multicast address has been built from an unicast subnet prefix or not and whether the address has been assigned permanently by IANA or not;
- the scope bits state the range of visibility for the multicast group. There are five relevant bits combination assigned by IANA: node-local, link-local, site-local, organization-local and global. This information is useful to routers in order to propagate properly the multicast packets to the appropriate set of hosts;
- the group id bits are user assigned, they are meant to identify a specific multicast host group.

The IPv6 protocol involves multicast addresses for everything in order to avoid the use of expensive broadcasts, as we can see in Table 1.5. Multicast addresses are used also for supporting address autoconfiguration, as we are going to see in Section 1.5.5.

Multicast address	Description
ff02::1	All nodes on the local network segment
ff02::2	All routers on the local network segment
ff02::5	OSPFv3 AllSPF routers
ff02::6	OSPFv3 AllDR routers
ff02::9	RIP routers
ff02::a	EIGRP routers
ff02::d	PIM routers
ff02::16	MLDv2 reports (defined in RFC 3810)
ff02::1:2	All DHCP servers and relay agents on the local network segment
ff05::1:3	All DHCP servers on the local network site
ff0x::c	Simple Service Discovery Protocol
ff0x::fb	Multicast DNS
ff0x::101	Network Time Protocol
ff0x::108	Network Information Service
ff0x::114	Used for experiments

Table 1.5: List of known IPv6 multicast addresses created by IANA (from Wikipedia)

1.5.4 ICMPv6 and the Neighbor Discovery Protocol

The ICMP [25] protocol has been used in IPv4 networks mainly to diagnose and test the connectivity between hosts, and a lot of vital tools like *ping* are based on its features, mainly the echo request/reply message types. The ICMPv6 [10] protocol has more responsibilities in IPv6, in fact it is the core of the Neighbor Discovery Protocol [22], the component responsible for:

address autoconfiguration stateless autoconfiguration of the network address, explained in detail in Section 1.5.5;

router discovery locating routers on the same link of the host;

address resolution mapping an IPv6 address with its correspondent Layer

2 address (essentially what ARP [24] does in IPv4);

duplicate address detection (DAD) discover whether or not an address is already in use;

reachability information (NUD) determine whether or not a node on the same link is reachable;

first hop redirect informing a node about a better first hop router (action performed by routers);

parameter discovery discovering of the link's parameters like MTU[8].

The Neighbor Discovery Protocol offers new services and re-implements some standard ones from IPv4, but it is important to understand that the big difference is in *how* it performs its actions: the NDP uses multicast instead of broadcasts if the underlying data link protocol supports it, easing the overall network load. For example suppose Node A wants to communicate to Node B, it holds its IPv6 address but it does not know the Ethernet MAC address. It then creates the Solicited Node multicast address of Node B appending to the reserved multicast prefix **ff02:0:0:0:0:1:ff00::/104** the last 24 bits of the IPv6 address of Node B and finally it sends a *Neighbor Solicitation* to that address and waits for the answer from Node B. The IPv6 protocol specification states that every node in the network must join its Solicited Node multicast group during the startup of its network connection, otherwise nothing would work. This mechanism prevent the use of heavy broadcast each time an address resolution occurs and it is an elegant solution because it is independent from the underlying data link network, using ICMPv6 instead of knowing the broadcast address of the data link layer (ARP in fact must be implemented for each data link layer type, like Ethernet).

[8]Maximum transmission unit: size (in bytes) of the largest protocol data unit that the link can pass onwards.

1.5.5 Stateless Address Auto Configuration

As mentioned in Section 1.5.1 one of the best practices in IPv6 is to reserve the last 64 bits of the address to the so called host identifier, in order to use it for features like autoconfiguration, one of the major changes from IPv4. To understand properly the Stateless Address Auto Configuration it is mandatory to understand how a host generates a 64 bit sequence that should have the strong property to be globally unique. The starting point is the Ethernet MAC address, that is composed of 48 bits and should be unique for a particular physical interface (for the sake of clarity we don't take into account interfaces belonging to Virtual Machines running on an Hypervisor, because obviously in this case the assumption is not true anymore). Subsequently if we add sixteen fixed bits to the MAC address we will obtain a globally unique host identifier. This procedure is the IEEE EUI-64 standard algorithm, this is a practical example of how it works:

- the starting point is an IEEE EUI-48 address, like 00:15:58:83:9f:08
- it is separated into two groups of bits: 001558 and 839f08
- they are joined together using the standard sequence of bits FFFF
- the address is formatted as stated by the IPv6 protocol: 0015:58FF:FF83:9f08

Here comes the brand new IPv6 features, the address autoconfiguration. Routers will have more features in IPv6 networks, like the so called Router Advertisements, a new ICMPv6 kind of packet responsible to announce a subnet prefix to hosts connected to a specific network. Let's go through an example: an host is connected to a network and it needs Internet connectivity. In IPv4 it has two possibilities: DHCP or manual configuration, that is someone tells the host which address to use. In IPv6 there is also the autoconfiguration option: if the host connected through the network receives a RA packet with a 64 bits subnet prefix it could generate a complete 128 bits IPv6 address using EUI-64, together with its default gateway, the router

responsible for the RA advertise. The DHCP is not so important anymore in IPv6 because of this new feature, that is meant to be not only an option but the standard.

Now we have more information to guess how a host boots up its network connection creating the link local address. As stated in Section 1.5.2 this kind of address is composed by a fixed subnet mask, namely **fe80::/64**, so a complete IPv6 address could be created appending the host identifier to the subnet mask. The same procedure is applied for the other types of unicast addresses, but in this case the host does not know the subnet prefix so it has to ask it to a third party source, the router.

1.5.6 IPv6 Header

The IPv6 header is depicted in Figure 1.3, it has a fixed length of 40 octets (320 bits) and it contains some new features from IPv4. First of all we can notice the two 128 bits address fields for the source and the destination, as we said this imply potentially thousands of IPv6 addresses for each square meter of the Earth! Each field has its own purpose, here a brief review:

- **Version** - contains the number 6, the version of the IP protocol used
- **Traffic Class** - a tag to assign different priorities to streams of packets
- **Flow Label** - a label to identify packets belonging to the same flow
- **Payload length** - self explanatory
- **Next Header** - the type of the next header, for instance the TCP or UDP (see Section 1.5.7 for a detailed introduction)
- **Hop Limit** - the equivalent of the TTL field in IPv4, namely the number of hops a packet could be forwarded by a node in the network
- **Source/Destination address** - self explanatory

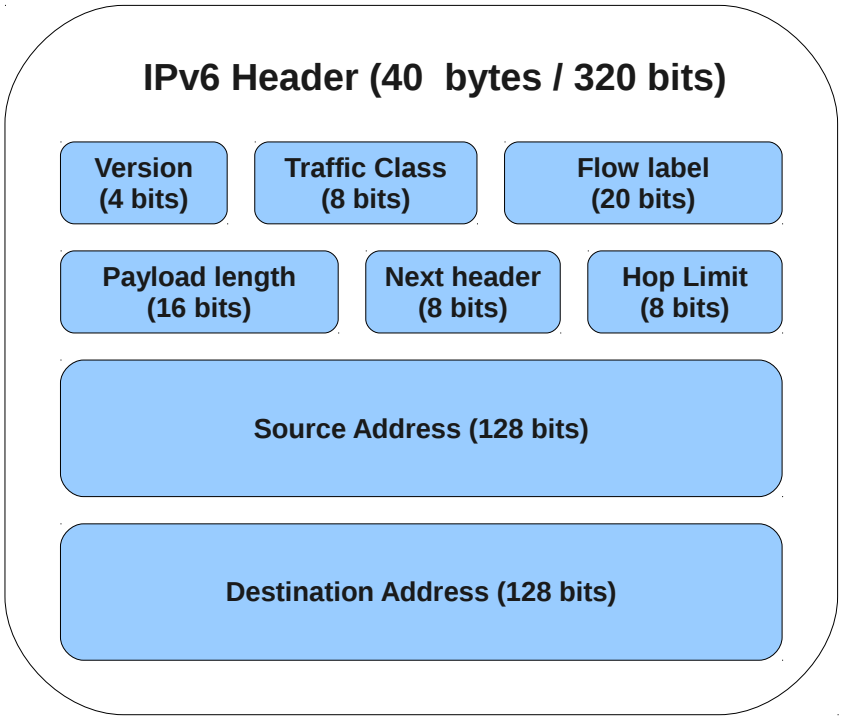

Figure 1.3: The IPv6 header fields

The IPv4 *Checksum* header field disappeared in IPv6, because the Layer 2 does the same work on frames. This change brings efficiency to routers that do not calculate anymore the checksum for each packet forwarded. Obviously the Layer 2 checksum will not spot Layer 3 router errors, but this will lead only to packet loss and retransmission due to common errors like address not existent and so on. Another header field disappeared is the IPv4 *Options*, but this will be addressed in Section 1.5.7.

1.5.7 IPv6 Extension Headers

The IPv4 header has a field called *Options*, a set of policies applied to the packet during its forwarding or to the end hosts participating to the communication. This field is not well designed for at least two reasons:

- **efficiency** - each time a router forwards an IPv4 packet it must read all the header fields, including the Options even if they concerns only end to end hosts;
- **modularity** - adding a new Options feature requires reserving a specific bit sequence.

The IPv6 protocol has been designed to address the above problems using an elegant software engineering solution, namely pointer jumping. As we can see in Figure 1.4, the IPv6 header has a fixed length of 40 bytes and it does not contains any Options field. Instead the *Next Header* field states the type of the next header the packet contains after the main one. The next header could be an extension of the main IPv6 header or it could be an upper layer header, for example the TCP one. This design permits routers to focus only to the main header during the packet forwarding and the extensions only when requested, for example in case of ACLs. The commonly used extension header types are:

Hop-by-Hop EH used to supports Jumbograms [7] or to support the operations of the IPv6 Multicast Listener Discovery (MLD);

Fragmentation EH used to support communications of fragmented IPv6 packets (in fact in IPv6 the traffic source must perform fragmentation, routers only forwards packets);

Destination EH used in IPv6 Mobility as well as support of certain applications.

The interested reader is invited to read the IPv6 RFC [11] for more information.

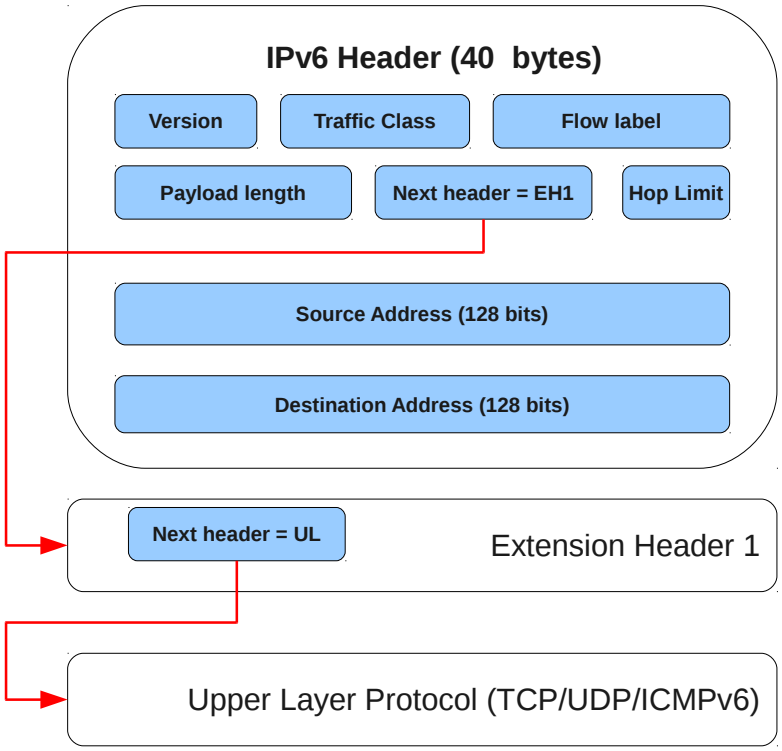

Figure 1.4: An example of the logical structure of one extension header for an IPv6 packet

Chapter 2

CINECA

"Dix," Case said, "I wanna have a look at an AI in Berne.
Can you think of any reason not to?"
THE NEUROMANCER

2.1 The Italian University Consortium

CINECA[1] is an Italian no profit consortium consisting of 54 Italian universities[2], the OGS[3], the CNR[4] and MIUR[5]; it is an high technology bridge among the academic world, the research, the industry and the public administration. Its activities cover:

- the support to scientific research through supercomputing and its applications, letting scientist to experiment the most recent HPC technologies together with extensive expertise and user support.

- Management systems, services and technical-training support to university administrative offices. Since the 1980's CINECA has supported Italian Universities in their main administrative areas like Students

[1] http://www.cineca.it
[2] by the time this thesis has been written, August 2012
[3] Istituto Nazionale di Oceanografia e Geofisica sperimentale, http://www.ogs.trieste.it/
[4] Italian National Research Council, http://www.cnr.it
[5] the Italian Ministry for Education, University and Research, http://www.miur.it

Management, Accounting and Human Relationships. This efforts led to the development of *U-GOV*[6], adopted by a large number of Italian universities. Moreover CINECA founded KION[7], a company focused on IT systems for student and learning services. This led to the development of another service called *ESSE3*, a Student Management System, which has also been adopted from almost all the Italian universities.

- Services for the Ministry of Education, University and Research. CINECA manages most of the online services related to the MIUR, using the GARR[8] network as communication infrastructure.

- Health Care Systems, more specifically the design and the development of IT systems and services in the health care and biomedical area, like various Web based system for the management of multicentric clinic trials and internal activities of various Health Care Organisations and scientific associations.

- Information and Knowledge Management Services, that is methods and techniques for the retrieval, management and analysis of data, information and knowledge.

2.2 The IT infrastructure

The CINECA infrastructure must support a wide range of services and technologies and above all it must be scalable and always efficient. The heart of such a big structure is obviously a reliable network, a keystone to build fast and strong Internet Web Applications.

The features of a well designed network should be the following:

- **scalability** - adding new subnets should be a straightforward operation and it must not slow down the rest of the preexisting environments;

[6]http://www.u-gov.eu
[7]http://www.kion.it
[8]the Italian Academic and Research Network, http://www.garr.it

- **reliability** - link failures should not compromise the network functionality;

- **speed** - the latency between hosts in the internal network or to the outside Internet should be as low as possible.

The CINECA network is mostly CISCO[9] based, at least in datacenter devices providing connectivity. It follows common best practices and some CISCO proprietary technologies to achieve the above design features. In the next subsections I will introduce the reader the most important best practices and CISCO add-on features used: Section 2.2.1 is about scalability and reliability in the network design, Section 2.2.2 is about managing the Data link layer efficiently, Section 2.2.4 is about network reliability and Section 2.2.5 is about the network traffic dispatching and separation. Finally Section 2.2.8 is about the CISCO Cloud Computing infrastructure adopted by CINECA.

2.2.1 Hierarchical Network Design

Networks design has evolved from flat to hierarchical topologies in order to let network architects to split functionality between layers to obtain modularity and flexibility. A typical enterprise network should be organized in four layers, as illustrated in Figure 2.1:

- **Access**: provides direct connectivity to hosts.

- **Distribution**: provides routing to the access layer, implementing policies for security and traffic loading, splitting networks into autonomous compartments.

- **Core**: implements the backbone of the network, a fast and redundant transport for the distribution layer.

- **Border**: provides the connectivity between the backbone and the Internet.

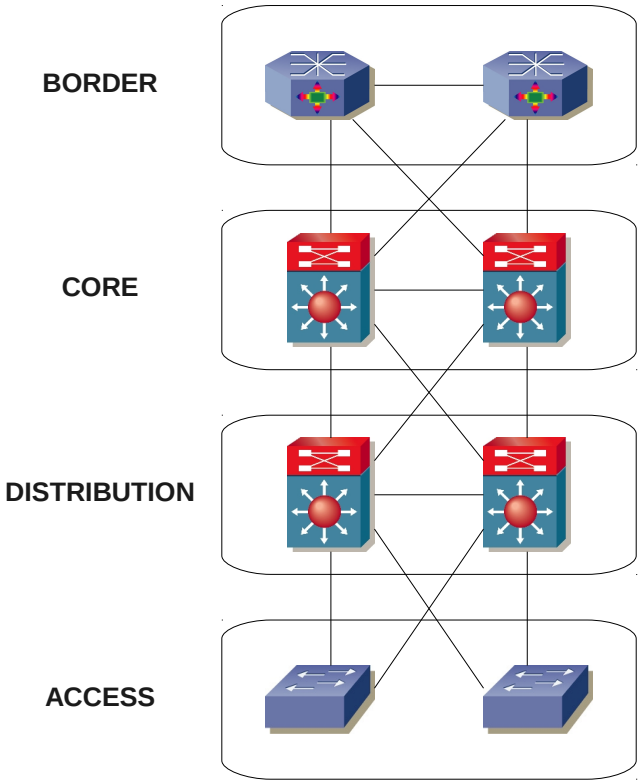

Figure 2.1: The Hierarchical Network Design for an Enterprise

The principal benefits of this approach to networks design are the following:

- **scalability** - network architects are allowed to replicate each module apart from the others as the network grows;
- **flexibility** - changes to a specific layer does not require change the others, especially for security and traffic management;
- **easier management and troubleshooting**: there is a clear distinction between Layer 2 switching and Layer 3 routing, that leads to more

[9]http://www.cisco.com

efficients network operations performed by engineers;

- **resiliency**: this is the most important feature, because it guarantees multiple redundant paths in the network for the same data flow. As the reader may notice in Figure 2.1, each layer has more than one path for the same starting point of a communication, a link failure should not compromise the overall network availability.

The reader familiar with the hierarchical network design may skip to Section 2.2.6 to learn how it is implemented into the CINECA network infrastructure.

2.2.2 Virtual Local Area Network

A Virtual Local Area Network (VLAN) is a Layer 2 technology standard [2] able to split the ports of a switch into multiple *broadcast domains*[10] without requiring any additional hardware. Moreover multiple switches connected together may share the same broadcast domains using a special packet tagging called *trunking*.

Suppose you have a Layer 2 switch and ten hosts connected to it, each one sharing the same broadcast domain. If you need to split the hosts into two subsets, each one containing five hosts, you will have to buy another switch, surely not a flexible solution. If the switch supports Virtual LANs you would simply create two separate broadcast domain called, for example, **A** and **B**, each one managing only the ports of the switch connected to its assigned subset of hosts. The VLAN A and VLAN B will communicate using the Layer 3 IP protocol, namely a router will join them.

A slightly more complicated example is the following: suppose you have two Layer 2 switches, ten hosts connected to each one and the same problem above, namely separate the total twenty hosts into two broadcast domains without rearranging the connection between hosts and ports. The two switches would be connected by a link used to exchange frames between

[10] A broadcast domain is a logical division of a computer network, in which all nodes can reach each other by broadcast at the data link layer. *Wikipedia, August 2012*

hosts of the same VLANs; each port would be tagged as *host* or *trunk*: the former indicates a direct connections to a host, the latter a connection between switches. The two ports connected to the trunk link must perform the additional work to tag each frame they send by its VLAN ID in order to keep the broadcast domains separated. This technique is called *trunking* and it is the standard that permits multiple switches to share the same set of VLANs.

2.2.3 Mapping VLANs to IP subnets

As stated in Section 2.2.2, two hosts belonging to different VLANs can communicate only through the IP Layer, therefore a mapping between VLANs and IP subnets is needed at this point. Let's go through an example: suppose to have a set of hosts connected to one or more Layer 2 switches, sharing the same broadcast domain and you need to split the network into two set of hosts, assigning to each set an IP subnet. The first thing to do is creating two broadcast domains using VLANs, for example calling them *A* and *B*, and then assign each one to an IP subnet. In our example we could map two IPv4 class C subnets to VLANs in the following way:

- **VLAN A** 192.168.4.0/24
- **VLAN B** 192.168.5.0/24

Finally a router must be connected to the switches in order to enable the IP routing. The router must have at least two interfaces (physical o virtual), one holding an IP address belonging to the range assigned to VLAN A and the other holding an IP address belonging to the range assigned to VLAN B (for example 192.168.4.1 and 192.168.5.1). The mapping between the VLANs and the IP subnets, two separate concepts belonging to separates Internet Layers, is achieved setting each interface of the router to its correspondent assigned VLAN. This is a simple operation available in all the recent router equipment, in fact CISCO implements it into its IOS operating system. A

graphical explanation of the differences between Layer 2 and Layer 3 concepts is depicted in Figure 2.2.

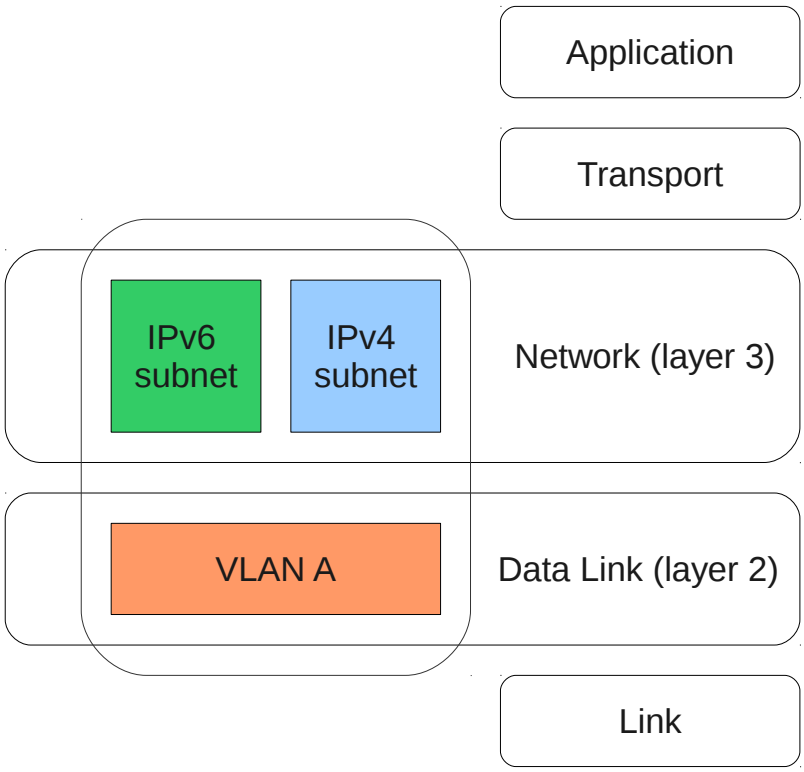

Figure 2.2: Example: mapping a VLAN to its correspondent IPv4/IPv6 subnets

2.2.4 CISCO Hot Standby Router Protocol

The Hot Standby Router Protocol (HSRP) is a CISCO proprietary protocol for managing redundant and fault tolerant default gateways assigned to IP subnets. This protocol is implemented often in the *Distribution* layer, where each IP subnet needs some redundancy in order to prevent losing connectivity due to a link failure. The protocol lets two routers share a virtual IP

address and a virtual MAC address, acting as a unique virtual router. They communicate with each other sending *hello* messages through IP multicasting in order to establish which one of them will answer to ARP/Neighbour Discovery[11] requests. One of the two routers needs to be set to *Active* while the other to *Standby* as starting point, and once a failure to the Active router occurs the other one will step in taking the place of the other one, transparently to the hosts of its IP subnet.

2.2.5 CISCO Virtual Router and Forwarding

The Virtual Routing and Forwarding technology permits to maintain two or more routing table instances on a single physical router, in order to separate traffic allowing the creation of virtual circuits for datagrams. This technology is useful when network engineers are requested to track and separate network traffic based on a set of rules. From an external point of view a VRF instance is a logical router completely separated from the other ones, although sharing the same hardware. For example, separate VRF instances can use the same IP subnet without conflicts; ISPs use this technology to implement Virtual Private Networks for customers using the same hardware without the need to use encrypted data channels, saving a lot of money buying less hardware. The network infrastructure of a datacenter could be implemented using a set of routers each one configured with the same set of VRF rules, allowing multiple clients to share the same hardware without conflicts or security flaws. Section 2.2.7 explains in detail how this technique is used to separate the network traffic in the CINECA network.

2.2.6 The CINECA Network Design

The CINECA network design follows all the best practices illustrated in the previous sections, as the reader can see in Figure 2.3. The picture may seem a little bit confusing and sketchy, but a closer look reveals that it follows

[11] Respectively for IPv4 and IPv6.

the hierarchical design principles illustrated in Section 2.2.1. Let's examine each network layer more accurately:

Internet Border This is the boundary between the internal network and the Internet, it must be a fast gateway and the first line of protection against malicious attacks at the same time. It is composed of routers, their first responsibility is to communicate with other routers through the Border Gateway Protocol [26] in order to establish dynamic routes to the Internet. CINECA uses a combination of peering agreement with third parties and announcements of its Internet prefixes from Provider Dependent and Independent sources. More specifically the *b01* router implements peering with the University of Padua, Kion and P.diMare, whereas *i01* and *i02* are directly connected to a GARR router for the academic traffic, and to Tiscali and Fastweb for commercial traffic.

Core The *c01* and *c02* routers are the backbone of the entire network, they dispatch all the traffic within the internal network and towards the Border layer. They do not implement any sort of filtering or security checks on datagrams, instead they offer a solid, redundant and fast service. The two routers use the OSPF routing protocol to dynamically find the best routes to reach the lower Distribution layer and the upper Border layer.

Distribution This layer does the real segmentation of the network, it splits all the traffic between separate compartments to different VLANs. As the reader may notice there are segments for the High Performance Computing systems, for the employees hosts, for the Production environments like clusters and farms, for the Virtual Private Networks for guests and for the Database hosts. Each segment communicates with the other ones through two routers implementing the HRSP CISCO technology (please see Section 2.2.4 for more details) in order to guarantee redundancy and resilience. The routers communicate through the OSPF routing protocol, announcing their subnets to the Core layer.

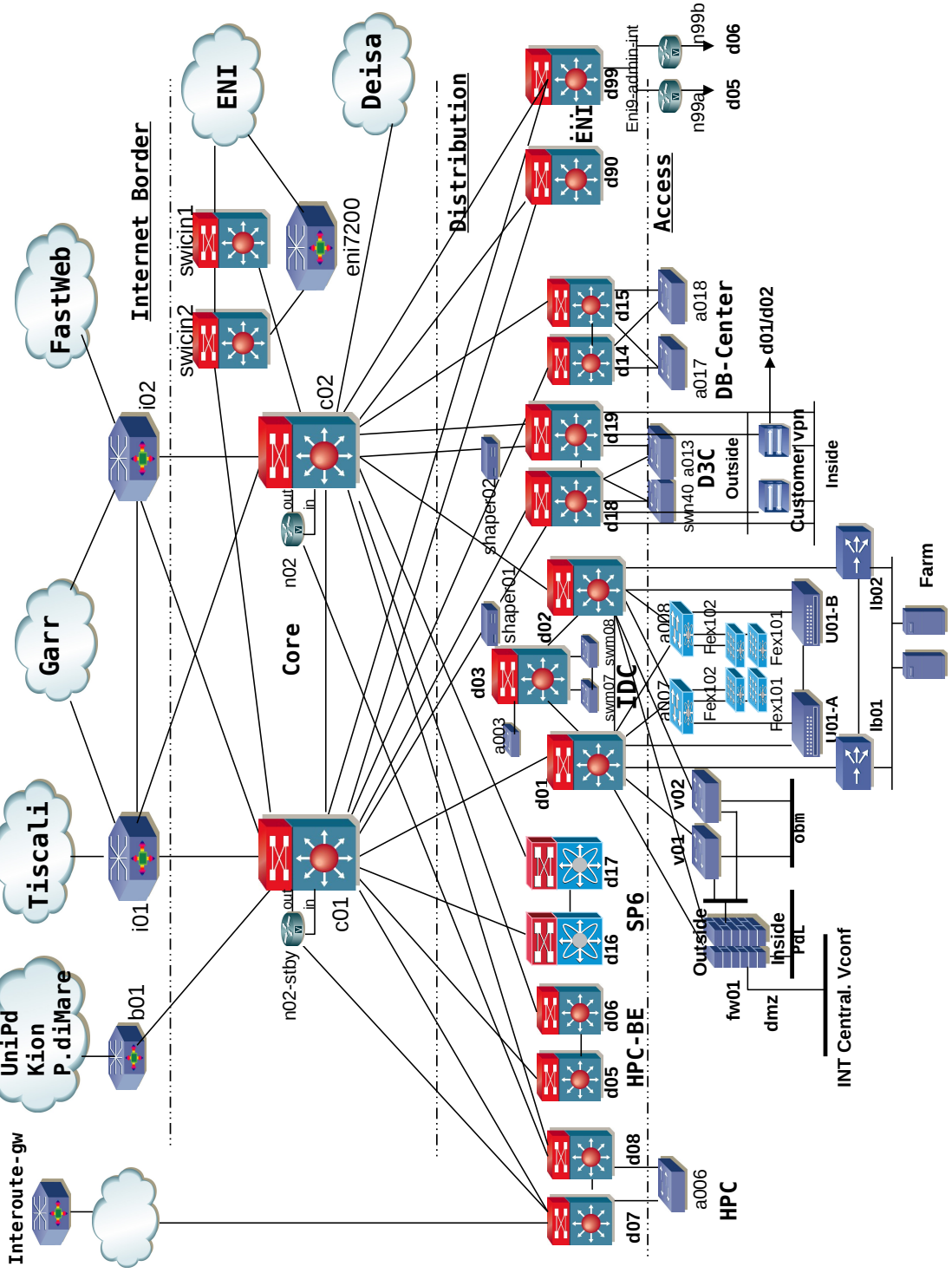

Figure 2.3: The Hierarchical Network Design at CINECA

Access This layer directly connects hosts to switches, it implements the VLAN segmentation and it joins them to their related Layer 3 IP subnets using the technique explained in Section 2.2.3.

The main difference from the scheme depicted in Figure 2.1 are the link connections between the Core and the Distribution layers: each distribution router is not strongly connected to the two core routers. The reason is the cost: each link between Distribution and Core is 10 Gigabit fiber, therefore it costs a lot in term of ports available on the routers and connection cables. Redundancy is achieved through the HSRP, each Distribution router monitors the availability of its links, and whenever one of them fails it will automatically decrease its HSRP priority (see Section 2.2.4) forcing the network data flow through the other Distribution router, therefore bypassing completely the failed link. Obviously this is a trade off, it does not achieve all the redundancy of a complete hierarchical design but it does guarantee a strong resiliency. Moreover the bandwidth between Distribution and Core is big enough to tolerate one link failure, whereas multiple ones could compromise the network availability.

2.2.7 The CINECA Autonomous Systems

CINECA is a consortium of universities, it offers Internet services and web hosting to third parties, so it needs some network traffic separation policies to maintain more efficiently its large infrastructure. In particular there are two big traffic categories flowing in and out the CINECA network:

- Academic traffic
- Commercial traffic

The Academic network traffic is generated from the Internet Services used by Italian universities and HPC community, it is managed though the GARR network and it has some traffic load policies. The Commercial traffic is associated to Internet Services and web hosting for the Private market, and it is

managed through the Fastweb and Tiscali networks. The naive solution to keep the network traffic separated could be to physically duplicate the network equipment into two Autonomous Systems and then connect them using a routing protocol like BGP. This solution is feasible for small enterprises, but it is clear that it is not scalable for medium and large infrastructures. As stated in Section 2.2.5 CISCO offers a technology called Virtual Routing and Forwarding to keep different routing table instances on the same network component, so it is feasible and easy to keep different *logical* infrastructures on the same physical hardware. It is now straightforward to guess how CINECA has implemented the network traffic separation: creating one VRF for the Academic traffic and another one for the Commercial traffic. Moreover each logical infrastructure created through a VRF is then associated to a separate Autonomous System ID. The WHOIS RIPE database is a good tool to investigate how the assigned IP subnets have been associated to different AS IDs. Let's start with the Academic VRF: it has been assigned to the IPv4 subnet **130.186.0.0/19** with assigned AS ID **AS137**, as the reader can see in Figure 2.4. A subsequent query to the Ripe Database for more information about the AS ID found shows us that the ID belongs to GARR (see Figure 2.7), in fact the subnet **130.186.0.0/19** is Provider Dependent, namely GARR act as Local Internet Registry for CINECA. This means that the real AS ID associated with the IPv4 subnet is private to GARR. Conversely the IPv4 subnet **130.186.64.0/18** is assigned to the AS ID AS3275 (see Figure 2.5), that is Provider Independent because of it comes directly from the RIPE RIR, as stated in Figure 2.6.
All the Autonomous Systems IDs stated above are related only to IPv4 subnets, but it is easy to extend them with IPv6 subnets; this let CINECA to skip the request for other AS IDs to the related autorities.

```
inetnum:        130.186.0.0 - 130.186.31.255
netname:        CINECA-NET
descr:          Consorzio di Calcolo Interuniversitario CINECA
country:        IT
admin-c:        ADF16-RIPE
tech-c:         AA107
status:         ASSIGNED PI
remarks:        GARR - Italian academic and research network
mnt-irt:        IRT-GARR-CERT
mnt-by:         GARR-LIR
source:         RIPE #Filtered

route:          130.186.0.0/19
descr:          CINECA-NET
origin:         AS137
remarks:        CINECA
mnt-by:         GARR-LIR
source:         RIPE #Filtered
```

Figure 2.4: The result of a WHOIS query submitted to the RIPE Database for the subnet 130.186.0.0/19 (August 2012)

```
inetnum:        130.186.64.0 - 130.186.127.255
netname:        CINECA-NON-GARR-NET
descr:          CINECA-NON-GARR
country:        IT
admin-c:        ADF1-RIPE
tech-c:         AA107
status:         ASSIGNED PI
remarks:        CINECA - Connettivita' Non Garr
mnt-by:         CINECA-MNT
source:         RIPE #Filtered

route:          130.186.64.0/18
descr:          CINECA-NON-GARR
origin:         AS3275
remarks:        CINECA - Connettivita' Non Garr
mnt-by:         CINECA-MNT
source:         RIPE #Filtered
```

Figure 2.5: The result of a WHOIS query submitted to the RIPE Database for the subnet 130.186.64.0/18 (August 2012)

```
as-block:       AS3209 - AS3353
descr:          RIPE NCC ASN block
org:            ORG-NCC1-RIPE
admin-c:        CREW-RIPE
tech-c:         RD132-RIPE
mnt-by:         RIPE-DBM-MNT
mnt-lower:      RIPE-NCC-HM-MNT
source:         RIPE #Filtered

aut-num:        AS3275
as-name:        ASN-CINECA
descr:          CINECA multi-homed Autonomous System
```

Figure 2.6: The result of a WHOIS query submitted to the RIPE Database for the AS3275 (August 2012)

```
as-block:       AS137 - AS137
descr:          RIPE NCC ASN block
org:            ORG-NCC1-RIPE
admin-c:        CREW-RIPE
tech-c:         RD132-RIPE
mnt-by:         RIPE-DBM-MNT
mnt-lower:      RIPE-NCC-HM-MNT
source:         RIPE #Filtered

aut-num:        AS137
as-name:        ASGARR
descr:          GARR Italian academic and research network
source:         RIPE #Filtered

role:           GARR LIR
address:        Consortium GARR
address:        Via dei Tizii, 6
address:        I-00185 Roma
address:        Italy
```

Figure 2.7: The result of a WHOIS query submitted to the RIPE Database for the AS137 (August 2012)

2.2.8 CISCO Unified Computing System

The CISCO Unified Computing System[12] (UCS) is a CISCO and VMWare[13] joint product that represent a flexible enterprise solution to build small and medium size Private Clouds[14]. The Wikipedia's definition[15] is more general:

> The CISCO Unified Computing System (UCS) is an x86 architecture data center server platform composed of computing hardware, virtualization support, switching fabric, and management software. The idea behind the system is to reduce total cost of ownership and improve scalability by integrating the different components into a cohesive platform that can be managed as a single unit.

The UCS deployed at CINECA is composed of the following components, as the reader can see in Figure 2.8:

- **D01 and D02 routers** - see Section 2.2.6;
- **a007 and a008 switches** - see Section 2.2.6;
- **UCS 6120 XP Fabric Interconnect** - 10G switches, they provide connectivity to the UCS's blade servers and the storage units using the Fibre Channel protocol over Ethernet;
- **UCS 6508 Blade servers** - CISCO blade servers;
- **UCS Storage** - the storage hardware components.

[12] `http://www.cisco.com/en/US/products/ps10265/technology.html`

[13] `http://www.vmware.com`

[14] Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party and hosted internally or externally, *Wikipedia, August 2012*

[15] `http://en.wikipedia.org/wiki/Cisco_Unified_Computing_System`, *Wikipedia, August 2012*

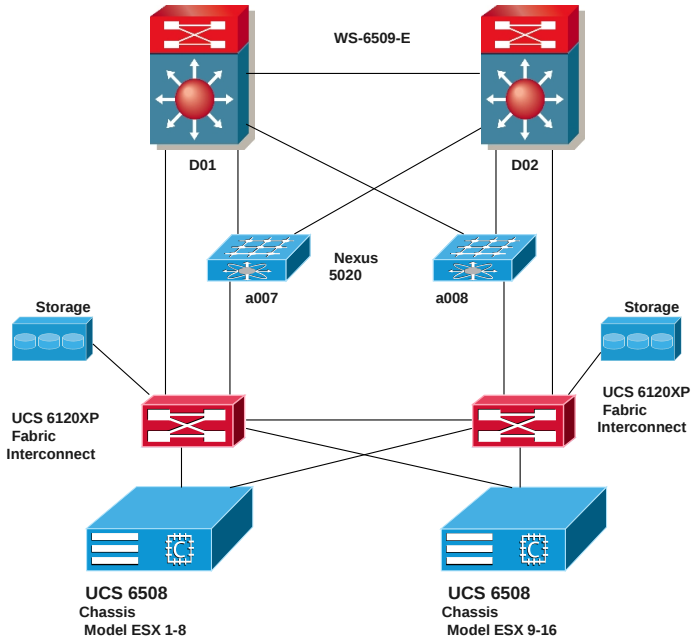

Figure 2.8: The network design for the CINECA UCS

This solution permits to deploy new hosts though virtual machines and connect them to a specific VLAN/IP subnet (as we said in Section 2.2.3 the two concepts overlaps) without requiring new hardware and networking components, obviously consistently with the maximum capacity of the UCS.

Part II

Towards a migration strategy

Chapter 3

The work plan

Everything should be made as simple as possible, but not simpler.
Albert Einstein

A well designed work plan should come after a study of the requirements followed by an analysis of the available tools and best practices. It should be clear that whatever the solution to the problem is, it will never be the panacea of the subject but only a complete working solution for the specific case analyzed. This is why the reader, before going any further, should read carefully all the motivations and details about our work plan.

This chapter is organized as follows: Section 3.1 collects the initial requirements for the migration, Section 3.2 shows a collection of solutions for different migration strategies and Section 3.3 states the complete work plan together with its main milestones.

3.1 Initial requirements

As stated in Chapter 2, the CINECA focus is to deliver reliable and efficient Internet Services. The keystone is changing the running IT infrastructure without interfering with the customers, in order to guarantee a productive interaction with the CINECA services. This is like replacing a gear to the engine of a running car: engineers can not stop and start the running sys-

tems to check correct configurations as they wish. The new features applied to a system must be tested into a separate environment from production, in order to apply once all the required changes affecting as little as possible the services continuity. The second big requirement is performing changes to the infrastructure without altering the user perception of the legacy software stacks, like incremented latency and less availability. Last but not the least, all the CINECA employee should keep working to their projects without any change in the network and Internet availability.

3.2 Migration strategies and solutions

The challenging part during the moving to a different addressing protocol is the need to work on each layer of the stack, because everything will be affected. The network layer obviously is the one requiring most of the attentions, but we should not forget about the application layer. The first thing the reader may think is: *Why would you do that? The Internet stack is designed to prevent this kind of problems, so it should be fine to migrate an application from IPv4 to IPv6 without changing a line of code!* Partially true, there are coding details to address properly in order to prevent application issues.

In this Section are grouped the most common best practices and available tools required to deal with the IPv6 migration with the requirements stated in Section 3.1. Section 3.2.6 presents the most common Application level problems, Sections 3.2.1 is about Network Address Translation and IP Tunnelling, Section 3.2.2 is about how organize and manage the migration, Section 3.2.5 is about the DNS changes required, Section 3.2.3 is about the Addressing Plan and Section 3.2.4 is about the necessary network security precautions to apply before going into production.

3.2.1 Transition to IPv6: Dual Stack, NAT IPv6 to IPv4 and IPv6 Tunnelling

Network Address Translation is a well established technique used in IPv4 to let private networks communicate with public ones saving precious public IPv4 addresses, using one public IP as multiplexer for multiple communications between hosts in the private network and the other hosts outside. The most advertised feature of IPv6 is the enormous address space, every device with a network interface will have a public Internet address in the near future, so why should we talk about NAT and IPv6? Imagine a scenario in which an enterprise network is segmented into IPv6 only and IPv4 only subnet islands: the latter could contain legacy hosts running old applications not designed for IPv6, whereas the former could contain hosts running applications on an IPv6 only environment in order to find issues with the new network stack before going into production. In this case network engineers need a tool able to *translate* the addresses from IPv6 to IPv4 and vice-versa transparently to the higher protocols. In this section we will analyze the NAT64[18][4] translation mechanism, focusing on connecting IPv6 only islands to the IPv4 Internet, as depicted in the example of Figure 3.1.

The NAT64 is a technique able to translate IPv6 headers to IPv4 ones, usually performed by a router managing an IPv6 subnet. There are two main type of NAT64:

Stateless The NAT64 router performs a one to one mapping between an IPv6 address in its subnet and an IPv4 one, usually embedding it into the IPv6 address through the use of specific algorithms (for more information please consult [18]). This approach scales very well because it does not keep connection states but its main drawback is the need of one separate IPv4 address for each IPv6. This means Stateless NAT64 can not be used in context in which the IPv4 address are few, like the actual IPv4 address exhaustion.

Stateful The NAT64 router performs a N to one mapping from IPv6 to

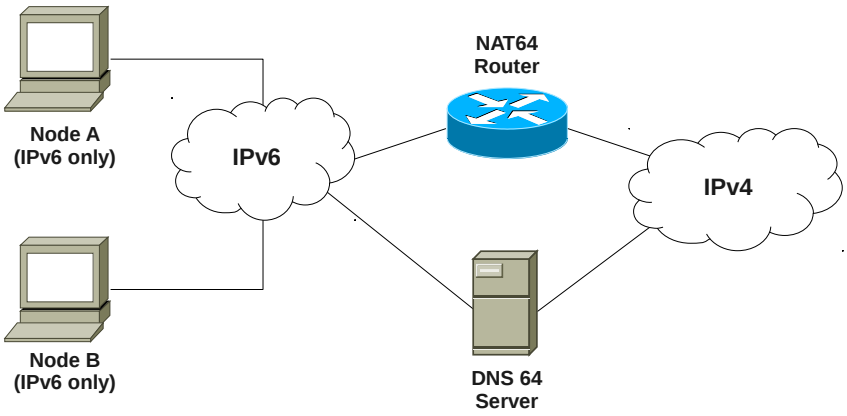

Figure 3.1: An example of NAT64 deployment

IPv4, but in this case it needs to keep the state of each ongoing connection as it happens today in NAT44[27]. This permits to have only one IPv4 address mapping an entire IPv6 subnet, but the drawback is that only an IPv6 host inside the subnet could establish a connection to an IPv4 host outside and not the opposite.

Moreover both of the above solutions must implement a third-party mechanism to translate DNS requests from the IPv6 world to the IPv4 world: in fact an IPv6 only subnet could make DNS requests for not existent AAAA records in the IPv4 cloud outside the NAT. The DNS 64[5] is a DNS resolution mechanism coupled with NAT64: it translates the AAAA record requests for a particular domain to the appropriate A record if the domain does not support IPv6, then it embeds the IPv4 address into an IPv6 one and returns it to the requestor. Let's go through a simple example, the one depicted in Figure 3.1: two IPv6 hosts behind a stateful NAT64 router. Suppose the following global IPv6 prefix as been assigned to the company owning the hosts: **2001:760::/32**. The network engineers could assign the prefix **2001:760:1:a:b:c::/96** to the NAT64 router, the IPv6 hosts will be manually configured with the addresses:

- Host A - 2001:760:1:a:b:c::1
- Host A - 2001:760:1:a:b:c::2

The DNS64 server and the NAT64 router would be configured as stated in Table 3.1, in dual stack IPv4/IPv6 mode in order to glue together the two protocols in the network.

Host	**IPv6 address**	**IPv4 address**
Router NAT64	2001:760:1:a:b:c::42	192.168.2.42
Server DNS64	2001:760:1:a:b:c::43	192.168.2.43

Table 3.1: Configuration example for a simple NAT64 deployment

An example of the TCP starting step initiated by Host A is depicted in Figure 3.2. As the reader may notice the keystone of the entire process is the trick used by the DNS64 server to embed an IPv4 address into an IPv6, that is appending to the tail of the IPv6 /96 network prefix the IPv4 address associated with the A record of the domain *example.com*.

Tunneling IPv6 into IPv4 is a technique able to encapsulate IPv6 packets into IPv4 ones, in order to join IPv6 subnets through an IPv4 path. This is really useful during the transition between the two protocols, because it permits to use IPv6 without the need to deploy it to the entire network. Imagine a company with multiple sites, each one experimenting the IPv6 protocol enabling the dual stack IPv4/IPv6 on a subset of hosts: it would be great to let them communicate through the IPv6 protocol without requiring to upgrade the entire network, using the pre-existing IPv4 transport. The most common suggestion is to use this kind of tunnels only during the start up phase of the migration if needed, because they are not contemplating by the IPv6 best practices as a permanent solution.

There are three main types of IPv6 to IPv4 tunnels: 6to4[8], ISATAP[28], Teredo[16].

In 6to4 each IPv4 globally routable address is used to build a unique IPv6 /48 subnet with prefix **2002::/16** appending the hexadecimal version of the

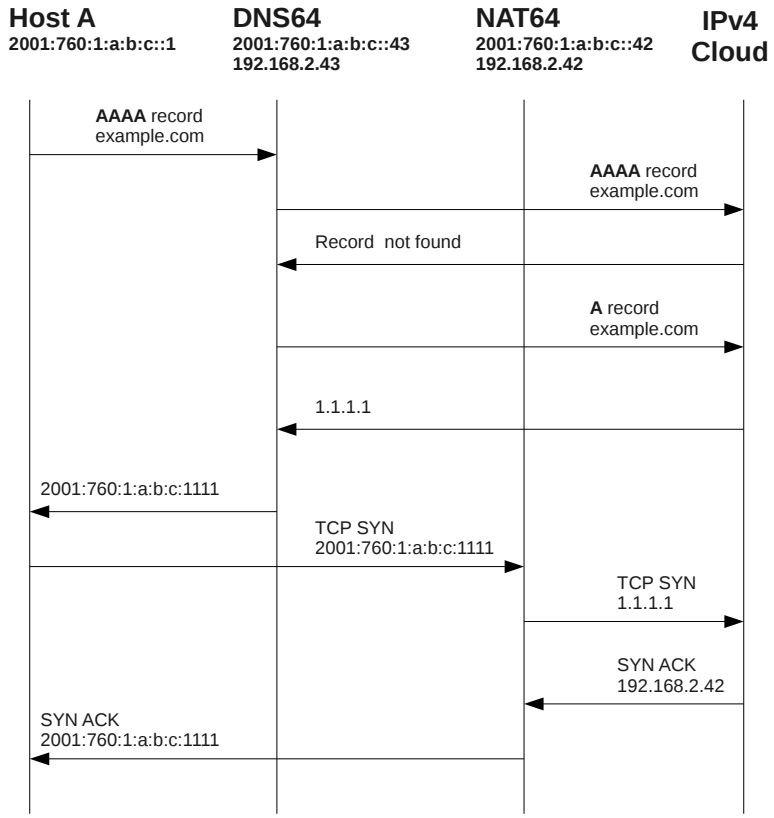

Figure 3.2: Example of a NAT64 scenario

32 bits IPv4 address to the reserved 16 bits prefix, this should guarantee the uniqueness of the 48 bits IPv6 prefix. Subsequently it would be possible to use the prefix to create different subnets or to build a complete IPv6 128 bits address, depending if there is the need to provide IPv6 connectivity to a group of hosts or only to one host. The typical use case is the deploy of 6to4 to two routers (the endpoints) holding each one an IPv4 global address in order to obtain two IPv6 /48 prefix and supplying IPv6 connectivity to the hosts they manage. This tunnelling technique needs also the presence of a well known relay in the outside Internet to work properly, the 6to4

RFC states that the IPv4 anycast address **192.88.99.1** and the IPv6 address **2002:c058:6301::** have been reserved by IANA for this purpose. A simple example of a 6to4 deployment is depicted in Figure 3.3. The reader should not confuse the 6to4 tunnelling technique with the 6in4[23] one, they share most of the implementation details but the former is more general and flexible respect to the latter.

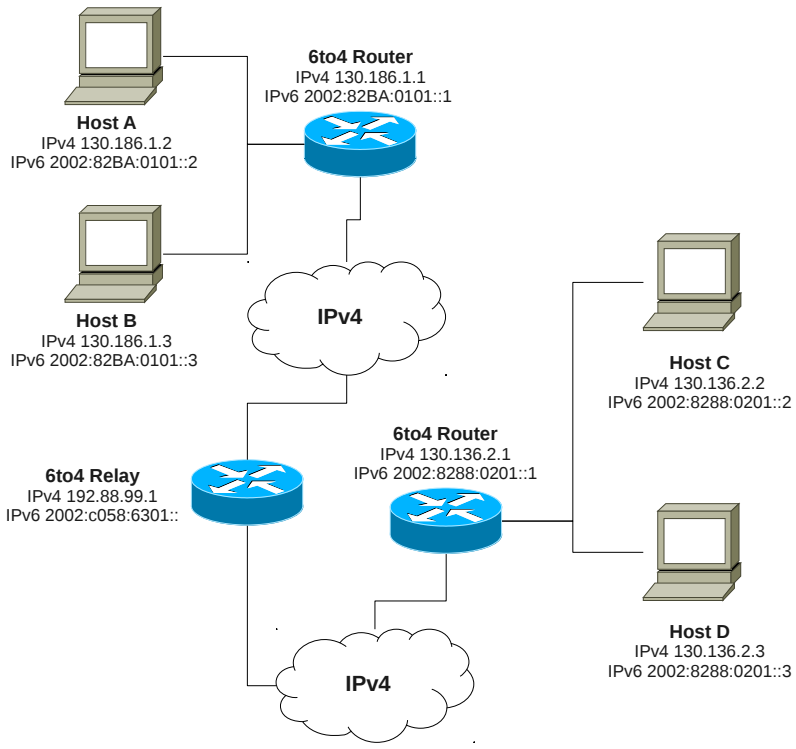

Figure 3.3: Example of a 6to4 deployment scenario

The Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) lets dual-stacked (IPv6/IPv4) nodes to communicate using the IPv6 protocol where the protocol is not directly supported from the network infrastructure, using the IPv4 protocol as non multicast/broadcast capable link layer for IPv6. This technique permits automatic tunnelling from IPv6 to IPv4 and it works

whether or not global IPv4 address are used. Using IPv4 as non multicast link layer interfere with IPv6 features like the Router Discovery, therefore ISATAP needs some external tools to work properly. For example in order to retrieve the list of available ISATAP relay routers it usually makes a query to the DNS name server configured with domain like *isatap.domain.com*, where *domain* is the local domain name. The major criticism about ISATAP is the need to have some support from Upper layer protocols like the DNS violating the Internet Stack principles.
Both 6to4 and ISATAP insert the IPv6 packet right after the IPv4 header setting the protocol type with the value 41, the reserved number indicating the presence of a tunnelling from IPv6 to IPv4. This technique is really efficient and offers a little overhead, but if an host is behind a NAT44 the tunnel will not work because it is not supported by the network translation mechanism (a host behind a NAT44 holds a private IPv4 address, not a global one as requested by the tunnelling protocols). Teredo offers a solution to the problem encapsulating the IPv6 packets into IPv4 UDP ones, thus relying on a transport protocol to establish a tunnel between IPv6 capable hosts. It needs a registration server holding a global IPv4 address and routers acting as relays to manage the connection between the source and the destination hosts: in this way both hosts can communicate behind NATs or symmetric firewalls without any connection problems. The hosts using a Teredo tunnel will build an IPv6 address starting with the prefix **2001::/32** combining it with the hexadecimal version of the Teredo server's IPv4 address and the NAT44's IPv4 address, the complete algorithm is described in detail into the Teredo's RFC. A Teredo deployment example is depicted in Figure 3.4 (example taken from [15]).

3.2.2 Top Down vs. Bottom Up

The most difficult thing to do before every big change is to choose where to begin. There are two main approaches in the networking world: top down and bottom up. The former is the optimistic one, it starts from the

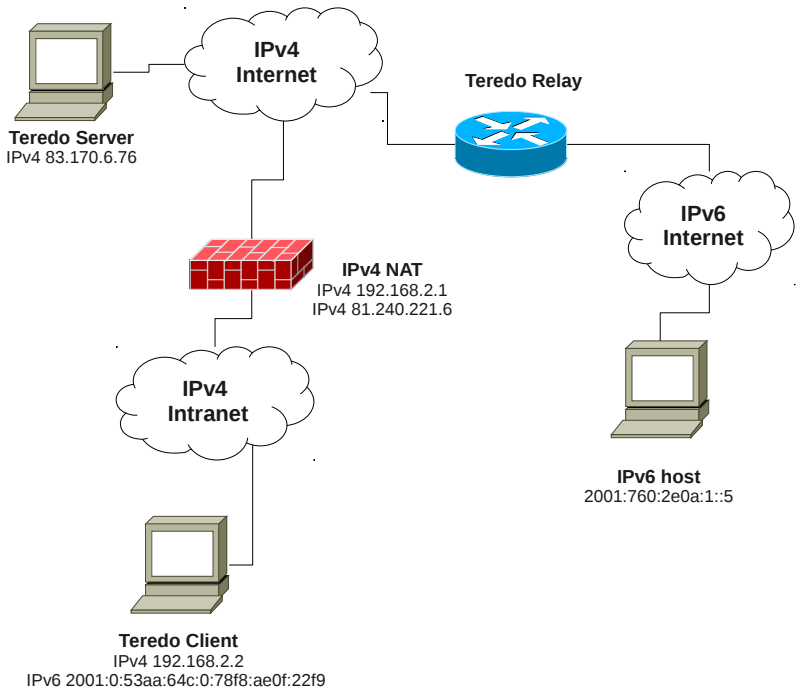

Figure 3.4: Example of a Teredo deployment scenario

network's backbone, where the Core routers work, enabling IPv6 using an interior routing protocol like OSPF. The Core is designed to be the skeleton of the network, it must be as fast as possible: it does not need security filters or packet inspection because this could lead to an overall network slowdown. Security is implemented in the Border layer and in the Distribution one, so the Core theoretically would be an optimum starting point for a migration to IPv6. Subsequently the upgrades should reach the Distribution layer, then the Access layer and finally the Border. This approach has the drawback to change a vital part of the network that could potentially create some network outages, so it is not recommended in the CINECA case of study (the reader should have read the migration requirements stated in Section 3.1). Moreover all the Internet services would be tested in the last part of the mi-

gration, leading to more possible failures, this time by the Application layer. The latter migration approach, the bottom up one, is more conservative: all the changes should start from a section of the network in the Access and Distribution layers, in order to test all the Network and Application components before enabling the IPv6 routing in the Core. Once everything has been tested and it can go to production, it is then enabled in the Border network layer in order to have a full IPv6 connectivity to the Internet using an exterior routing protocol like BGP.

3.2.3 Addressing Plan

The first mandatory step to set up a network that will be connected to Internet is obtaining an IP address range, choosing from the following types:

Provider Dependent a Local Internet Registry (LIR) owning a very large IP prefix assigns part of its subnets to a third party entity, usually because they belong to the same organization or they have a business in common. For example CINECA manages a Provider Dependent IPv4 /19 prefix on behalf of the GARR LIR because they have different roles in the same business, namely the Italian Academic world. The only drawback of a PD IP prefix is that the third party entity does not own it, meaning it has to subscribe some traffic policies with the owner. In the GARR/CINECA example CINECA has the restriction to use the GARR Network and its related IPv4 prefixes only for the academic and HPC traffic, leaving outside the one generated by commercial applications sold by CINECA.

Provider Independent the network owner can obtain an IP range directly from a Regional Internet Registry (RIR) like RIPE, buying the address range and therefore owning it. This solution is more expensive than a Provider Dependent one but it does not have traffic restriction of any kind with an external entity.

As stated in Section 2.2.7, CINECA manages both PD and PI IPv4 prefixes and the plan is to do the same for IPv6.

Once an IP prefix has been obtained by the network owner best practices suggest to partition the available subnets in a hierarchical tree in order to manage easily the creation of new subnets on the everyday work. The subnet hierarchy should start with few macro traffic categories and subsequently split them into fine grained subnets until reaching the hosts addresses (the leafs of the tree).

3.2.4 Security measures

Network security could be splitted into three main categories: Data plane protection, Control plane protection and Management plane protection. Each one has its own best practices and tools able to mitigate or prevent attacks from malicious users, focusing on different aspects of the network:

Management plane protection concerns the way a user gain access to the configuration commands of routers or switches, physically connecting a cable or remotely through Secure Shell (ssh), Telnet or SNMP. Securing the access to network components is vital for the overall security of the system, the first priority of this layer is to let only a restricted and authorized number of users to change the configuration of network components.

Control plane protection concerns all the network traffic destined to the router, for example:

- the packets related to routing protocols like OSPF, BGP, EIGRP, etc. . . ;
- the ICMP packets generated by a ping directed to a router's IP address;

- the SNMP packets generated by monitoring tools like Nagios[1] directed to a router's IP address.

Modern router hardware has dedicated CPUs to process this kind of traffic, often forged packets are used to induce Denial of Service to network components overloading their CPU capacity. The most common prevention is to create different queues to hold incoming packets following some categories (for example monitoring, routing, icmp) and apply ad-hoc ACLs to each one in order to drop unnecessary network traffic if it exceeds some threshold.

Data plane protection concerns the network packets generated by hosts and forwarded by network components among the network. A malicious user could compromise one host and use it to gain access to the other ones using standard attacks like Man in the Middle or packet forging. Therefore routers should mitigate this problem inspecting packet headers and dropping suspicious traffic.

In a hierarchical network design each layer should be configured to address the security concerns stated in the above list, in particular the Distribution layer and the Border layer.

3.2.5 Domain name system

The Domain Name System[20], or DNS, is one of the most important Internet services: it couples IP addresses with mnemonic names in order to abstract Internet users from the addressing scheme. The DNS protocol can hold various types of record and the IPv6 protocol has one of them reserved, namely the *AAAA* record. For example, if we use the *dig* utility asking to the DNS resolver what IPv6 address the domain *google.com* has we obtain:

```
dig AAAA google.com
```

[1] `http://www.nagios.org`

```
;; QUESTION SECTION:
;google.com. IN AAAA

;; ANSWER SECTION:
google.com. 286 IN AAAA 2a00:1450:4002:802::1002
```

A company migrating to IPv6 must add the AAAA records for all of its Internet services listening to an IPv6 address in order to have them reachable from the outside Internet. This means not only adding records to the proper DNS zones, but also redefining all the automation software responsible to manage those zones (it is rare to let engineers do this kind of work by hand in a medium/large infrastructure). Moreover the hosts running the DNS name servers and resolvers have to migrate to the IPv6 protocol in order to answer to DNS queries made by hosts communicating with IPv6: this is a very delicate step because it requires a lot of testing before reaching the production, the DNS is a mission critical service that could compromise the entire Web business of a company in case of outages.

3.2.6 The Application layer

In a perfect world this section would be unnecessary because from the software development point of view the Internet stack should abstract all the complexity of the communications between two or more hosts from the Application layer, leaving all the responsibilities to the Network and Transport ones. This is partially true, but as we will see there are a lot of details to take care in order to migrate an application to the dual stack IPv6/IPv6 without any issue.

Suppose you are writing an application in the C language that will be executed on a dual stack IPv4/IPv6 Unix host connected to Internet. This application will consume services from other hosts using Unix Network Sockets and the Domain Name System. For instance the application will be required

to connect to the port 80 of *www.example.com* using a Unix socket without knowing which IP protocol version will be available. One solution is to use the *getaddrinfo* function in order to translate the domain *www.example.com* to a list of canonical host names for the domain and then choosing one of them to establish a socket connection. But if the domain is available in both IPv6 and IPv4, the list of canonical host names will be composed by some names resolving in IPv6 addresses and other ones resolving in IPv4 addresses, which one should have the priority? Should the operating system be responsible to sort them following some policies or not? This problem might seem secondary but it could affect the application's overall responsiveness, making it barely user friendly. Suppose the host owns for the same physical interface a global IPv4 and an Unique Local IPv6 address (see Section 1.5.2): if the application tries to establish a connection to *www.example.com* using an IPv6 socket it will surely fail with a socket timeout, because the address assigned to the interface is not allowed to cross the borders to the Internet. Subsequently if the application is well designed it should try to establish an IPv4 socket to one of the IPv4 canonical host names for the domain but this behaviour will lead to a poor user experience because of the added latency generated each time the hosts tries to connect to a dual stack domain like *www.example.com*.

3.3 The plan

The previous sections illustrate how many different approaches are available to organize a migration to a new network technology like IPv6. Obviously there is no optimal solution for every situation, the one chosen by CINECA has been influenced heavily by its actual[2] infrastructure and by the requirements stated in Section 3.1.
The work plan is built around the following choices:

- dual stack IPv4/IPv6 on every host;

[2] August 2012

- bottom up strategy.

Choice fell to the dual stack because is a long term solution and there is no need for NAT or Tunnels, whereas the bottom up strategy is a conservative way to test the IPv6 stack far from the production environment. Best practices suggest to set up a test pilot composed by few hosts and some routers/switches in order to experiment various hardware/software configurations without the risk of creating issues to the rest of the infrastructure.
The work plan is organized as following:

- **First part**: experiment IPv6 into a testing environment

    1. Create the Addressing Plan's skeleton, identifying the main network traffic macro areas and temporary subnets for the IPv6 lab.
    2. Setup of a the IPv6 lab, using five Virtual Machines (see Section 2.2.8) instantiated with different Operating Systems (Debian, Windows 7 and Windows 2008 R2). The hosts are separated into two main sets, Clients and Servers, corresponding to separate VLANs and IPv6/IPv4 subnet, in order to simulate a client/server environment. The correspondent Distribution routers are configured as stated in Section 2.2.3, their purpose is to join the VLANs using IP routing.
    3. **First Milestone**: IPv6 communication between hosts belonging to different IPv6 subnets (mapped on different VLANs).
    4. First Hop security, namely applying Data/Control/Management plane protections to the Access and Distribution Layers.
    5. Manually create and fill the DNS zone *ipv6.cineca.it* with the IPv6 addresses of all the Lab hosts.
    6. Testing client/server software stacks deployed to the Client/Server VLANs with the IPv6 protocol.
    7. **Second Milestone**: IPv6 communication between applications tested.

8. Routing between the Distribution and the Core Layers.
9. Border security, namely applying Data/Control/Management plane protections to the Border layer.
10. Enabling the BGP protocol to the GARR network, connecting the VLANs to the Internet.
11. **Third Milestone**: testing environment connected to Internet through native IPv6.

- **Second part: deploy IPv6 to production**

1. Find some early adopters for production services to deliver in native IPv6.
2. **First Milestone**: IPv6 enabled production environment working.
3. Expand the IPv6 capability to the whole network.
4. Adapt the automation software to support IPv6.
5. **Second Milestone**: management process IPv6 compliant.
6. Dual stack support enabled for every new server deployed.
7. Migration to the dual stack and dismission of IPv4-only servers and services.
8. **Third Milestone**: dual stack enabled to the whole infrastructure.

The first part of the plan is the core of this thesis, all the work done is described in detail into Chapter 4, whereas the second one is marked as *future work* but it is described in detail in Chapter 5.

Chapter 4

The IPv6 Lab

Each new user of a new system uncovers a new class of bugs.
Brian Kernighan

Every new technology like IPv6 should be studied and tested in a separate environment in order to prevent deployment issues to reach the production. This is why I spent most of my time at CINECA working in a small environment called the IPv6 Lab in order to experiment the IPv6 protocol. In Chapter 3 we saw an overview of the work to do, meanwhile in this one we are going to focus on how that work has been done giving a deeper look to problems encountered and technical solutions chosen.

This chapter is organized as follows: Section 4.1 describes the design and the implementation of the CINECA IPv6 Lab, Section 4.2 and Section 4.4 describe all the tests performed in the Network and Application layers, finally Section 4.3 is about the IPv6 Security vulnerabilities and their related countermeasures.

4.1 Design and implementation

4.1.1 The design

In order to prevent deployment issues to reach production it is necessary to recreate in a smaller scale its important components: not only the ecosystem around the server machines (like farm and clusters) but everything has to work everyday, like the employees workstations. This means testing all the Internet stack, from the network routing to the operating systems and their applications. Obviously it is not feasible to test all the possible configurations, but only the most important as stated in the following list:

Network components The IPv6 protocol must be tested from different network viewpoint, namely same link traffic (same broadcast domain) and routing through different broadcast domains (for example different IP subnets mapped on VLANs). This is necessary not to test if the IPv6 protocol works but how well the actual routers support and handle the new kind of traffic. Moreover IPv6 introduces new features like host address autoconfiguration, so it is important to know whether they are a good solution to deploy or not.

Operating systems All the major operating systems advertise their IPv6 compatibility, but some real world tests must be performed. The ones supported by CINECA into its production environment are Windows 2008 Server R2, Windows 7, GNU/Linux Debian and Red Hat Linux.

Application stacks The Apache Web Server, the IIS Web Server, Apache Tomcat and Red Hat JBoss are ubiquitous server applications at CINECA, therefore they must be tested with IPv6. Sample web applications would help to test how an Application stack handles multi-protocol requests. Let's make an example: suppose we need to deploy a Java Web Application, a Model-View-Controller that communicates with a specific web service running on another host using IPv4 and renders the result on a web page. Suppose also the former host has been

configured with Tomcat binded to port 80, accepting both IPv4 and IPv6 traffic. What does it happen when an IPv6 HTTP request hits the port 80 of the server? The optimistic answer is straightforward: the HTTP request would be extracted from the TCP/IPv6 transport, then processed by the Java MVC framework using the IPv4 stack if necessary, then an HTTP response would be created and sent by Tomcat through the TCP/IPv6 transport. This behaviour is what we would expect but it must be tested in a real world environment. Moreover most of the web applications interact with the Domain Name System in order to resolve domain names, they have been designed to use only IPv4 as Network transport, so what happens when they need to choose between IPv4 and IPv6?

Security The network security must also be tested to find suitable rules and policies for Management, Control and Data Plane protection for each layer of the CINECA infrastructure (namely Border, Core and Access/Distribution).

The first concept to model in the IPv6 Lab is the client/server interaction, so it is necessary to dedicate some hosts as servers and some others as clients. They need to communicate using the Network layer to simulate an Internet connection, therefore they should not be deployed on the same broadcast domain (i.e. same VLAN) but a router should join a separate VLAN/Subnet IPv6 dedicated to servers with another one dedicated to clients.

4.1.2 The implementation

As stated in Section 2.2.8, CINECA has recently deployed the CISCO UCS infrastructure: it runs the VMWare virtualization software that permits the creation of Virtual Machines managed by a hypervisor, together with network components virtualization (mostly switches). This approach is more flexible compared to the physical solution, especially when the hosts have limited temporal lifetime, as the one dedicated to the IPv6 lab. All

the operating systems needed were ready to use as Virtual Machine images, the set up of a host is really fast because most of the configurations have been already done prior the creation of the image. The goal was finding a consistent way to deploy both IPv6 and IPv4 stacks on the same host, we shaped the Lab using four different IP subnets:

- an IPv6 subnet for the server hosts;
- an IPv4 subnet for the server hosts;
- an IPv6 subnet for the client hosts;
- an IPv4 subnet for the client hosts.

From the Layer 2 point of view two VLANs are needed, one for the server hosts and one for the client ones. This configuration has been pushed to the Access and Distribution layer; for a complete description of the network components used the reader should go to Section 4.2.2.

4.2 The Network layer

The last part of Section 2.2.7 describes how the CINECA IPv4 subnets assigned by GARR and RIPE (respectively Provider Dependent and Independent sources) are mapped to Autonomous Systems Ids. The migration to IPv6 has requested the acquisition of two IPv6 Internet prefix:

Provider Dependent : 2001:760:2e0a/48 assigned by GARR;

Provider Independent : the request is[1] still in progress, CINECA has become a Local Internet Registry (LIR) appointed by the RIPE RIR, the IPv6 prefix should be available soon.

For obvious time restrictions this thesis has been focused only on the IPv6 prefix assigned by GARR, but all the results obtained by the IPv6 Lab will be extended to the RIPE IPv6 prefix very easily.

[1] October 2012

4.2.1 The addressing plan skeleton

As stated in Section 3.3, the first step of the migration strategy is to define the skeleton of the IPv6 addressing plan: it helps to structure the network and bring order to the initial chaos. Obviously the IPv6 Lab alone would not need such a plan, it could be possible to pick two IPv6 subnet ranges from the /48 prefix assigned and start working, but the drawback is that it could lead to confusion in the next steps of the migration.
The address plan skeleton has been shaped around the CINECA hosts taxonomy:

- servers like Oracle Data Base Management Systems (DBMS) that need to retrieve updates from the Internet and incoming traffic only from host in the internal network;
- servers like Apache, JBoss and the HPC's Job Acceptors that need to receive and send traffic from the outside Internet;
- clients like the employee's workstations that need to communicate with the outside Internet and receive traffic from it with some restrictions.

The skeleton created is depicted in Figure 4.1:

1. the /48 prefix has been partitioned into two /49 subnets, a *private* one and a *public* one. The former is a set of global IPv6 addresses coupled with some ACLs in the Border layer allowing internal hosts to open TCP/UDP connection towards the Internet but not the opposite, meanwhile the latter is a collection of global IPv6 addresses allowed to handle TCP/UDP connections in both flows of communication. An example for TCP connections is depicted in Figure 4.2.
2. Each /49 subnet has been partitioned into other two, related to the CINECA's macro categories HPC and ICT.

We choose to assign Global IPv6 addresses to each host of the network, avoiding the Unique Local ones, because for practical reason is more feasible

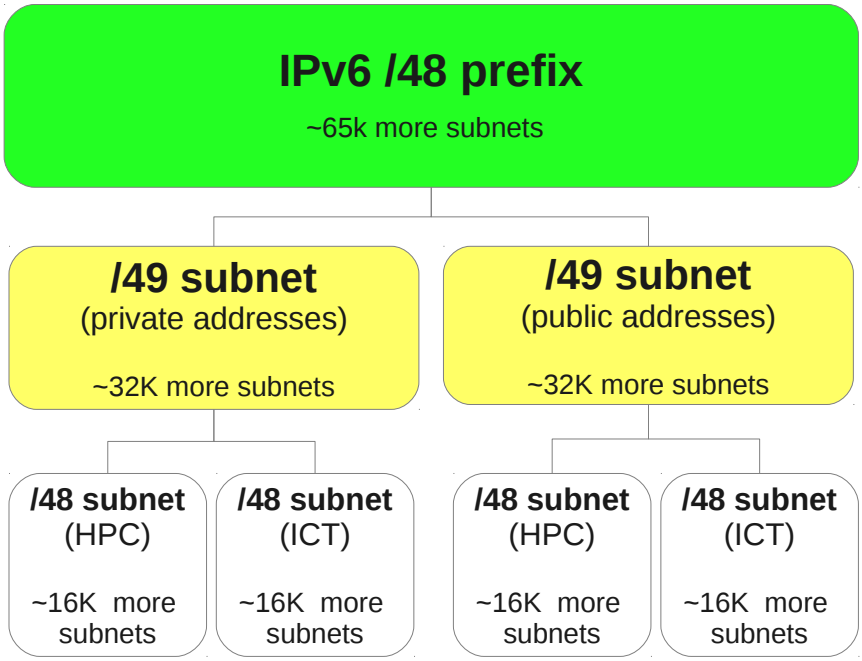

Figure 4.1: IPv6 Addressing plan skeleton

to make every host potentially Internet reachable and restrict its visibility using appropriate subnets shaped by ACLs. The server hosts have been assigned to the public /49 subnet to let Internet clients to establish new TCP/UDP connections, meanwhile the client hosts have been assigned to the private subnet to protect them from unnecessary exposure to the Internet traffic.

4.2.2 The IPv6 Lab network

For practical reasons only the network components required to let the IPv6 Lab's hosts communicate with each other and with the Internet hosts outside CINECA have been chosen to run the IPv6 stack, the other ones will be upgraded in subsequent steps of the migration. The CINECA UCS 2.2.8 is deployed in the network depicted in Figure 4.3, that is basically the main

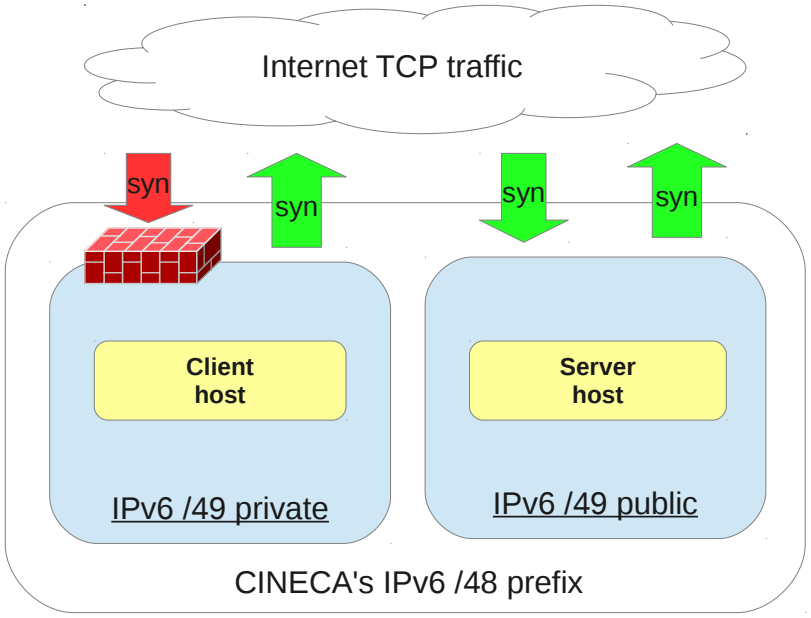

Figure 4.2: Example of how a TCP SYN request is handled by hosts in different IPv6 subnets.

skeleton of the overall network infrastructure. In particular here is the list of the network components together with their vendor details:

i01/i02 : CISCO 7604 edge routers

c01/c02 : CISCO 6509 multilayer switches

d01/d02 : CISCO 6509 multilayer switches

a007/a008 : CISCO Nexus 5020 Layer 2 switches

As stated in Section 2.2.7 the network routers *i01, i02, c01, c02, d01, d02* use the VRF technique to let multiple routing table coexists on the same hardware, so we chose to test the IPv6 stack only in the Academic

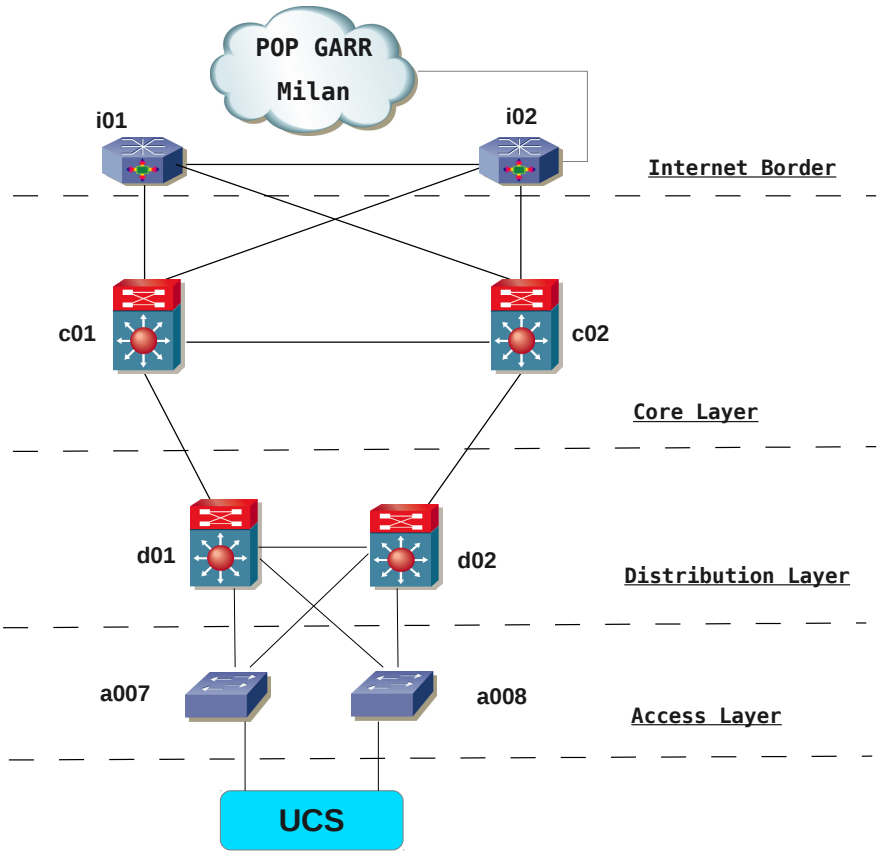

Figure 4.3: Subset of the CINECA network infrastructure tested within the IPv6 Lab

instance given that during my internship[2] the only available IPv6 Internet prefix was the one provided by GARR (see Section 2.2.7 and Section 4.2.1).

4.2.3 DNS and IPv6

Adding a new IP address type means modify also the DNS protocol in order to support the new traffic flow. In particular there are two DNS components to take care of:

- Resolvers

[2]July 2012 to October 2012

- Name Servers

The former one is the server application used by clients to resolve recursively domain name translations, meanwhile the latter manages the DNS zones in its delegation tree and answers to the resolvers queries. For example, CINECA has two resolvers used by each host in the internal network to make DNS queries, and two name servers (primary and secondary) managing the *cineca* domain for the *com* and *it* top level domains. The IPv6 traffic must be handled on two sides: the first one is DNS requests encapsulated in IPv6 packets, the second one is DNS AAAA record requests to translate domain names to IPv6 addresses. We decided to proceed trying to change as little as possible the CINECA DNS, namely leaving the CINECA Resolvers and Name Servers serving only IPv4 traffic and creating an ad hoc IPv6 zone called *ipv6.cineca.it* to deal with the IPv6 Lab hosts' AAAA records. The DNS hosts will be migrated to the dual stack IPv4/IPv6 only when the overall deployment will be at a mature stage to prevent service outages.

4.2.4 DHCP and autoconfiguration

The DHCPv6 protocol [13] is the IPv6 version of the famous Dynamic Host Configuration Protocol used in IPv4, it is used to assign IPv6 address to hosts following some policies. The big difference with its IPv4 counterpart is that it does not deliver routing information like the default gateway, because this task is already performed by Router Advertisement in the Neighbor Discovery Protocol. Moreover an IPv6 host could communicate immediately within its broadcast domain simply autoconfiguring its link-local address, meanwhile in IPv4 the host is either manually configured or dynamically through DHCP. This behaviour is an IPv6 design choice to prefer the use of host autoconfiguration, therefore the DHCPv6 should be bound to deliver only information not contained in a Router Advertisement like DNS resolvers and NTP servers.
We choose to statically configure the IPv6 stack for the server hosts (Global IPv6 address, default gateway and DNS resolvers for each host interface),

whereas use the SLACC for the client hosts using as DNS resolvers the ones configured for the IPv4 stack. This allows CINECA systems engineers to have a fine grained control over the servers deployment, meanwhile delivering IPv6 connectivity to clients simply enabling router advertisements (Linux and Windows are configured by default to accept router advertisements and autoconfigure their IPv6 stack with SLAAC). Moreover DHCPv6 client implementations are not mature both in Linux and Windows, we prefer to test the host autoconfiguration while waiting for a more mature implementation of the DHCPv6 protocol from OS vendors.

4.2.5 IPv6 Privacy Extensions

A host using IPv6 address autoconfiguration couples the subnet prefix announced by a Router Advertisement with a self generated host identifier, most of the time following the EUI-64 protocol. It is easy to find the original MAC address of the host interface simply applying the EUI-64 protocol in reverse: this means a global autoconfigured IPv6 address is easily trackable, a big privacy concern for an Internet user. IPv6 Privacy Extensions [21] came to the rescue: the global autogenerated address is supported by a temporary one used only as *source* address for the outgoing IP packets. This temporary address has a fixed limited lifetime, set by the OS and configurable by the user. It is different from the original global one in the host identifier, created using pseudo-random number generators (the actual implementation depends by the OS) instead of the MAC address. Windows 2008 Server and Windows 7 are shipped with Privacy Extensions enabled by default, meanwhile the GNU/Linux Debian distribution (and most of the others) needs specific configuration commands to enable them. We choose to set Privacy Extensions disabled on all the *server* hosts, leaving CINECA users to decide for the *clients* hosts. The drawback of this approach is adding entropy in the *client* hosts monitoring, but at the moment this is the best trade off between the need of control by systems engineer and the users privacy.

The Windows OS needs the following commands to disable Privacy extensions:

```
netsh interface ipv6 set global
    randomizeidentifiers=disabled store=persistent
```

The GNU/Linux Debian distribution needs the following commands:

```
sysctl -w net.ipv6.conf.all.use_tempaddr=0
sysctl -p
```

4.2.6 Routing between Network layers

The CINECA network is built around the concept of VRF (see Section 2.2.7), namely the technology used by routers to manage more than one routing table on the same device. The OSPF protocol joins Distribution, Core and Border layers redistributing the IP subnets to all the routers of the network: in order to work a router must handle properly the OSPF packets to maintain the VRF instances separate. Unfortunately the CISCO IOS firmwares for the routers listed in Section 4.2.2 are not ready at the moment[3] to handle OSPF and VRF, but the will be ready only from March 2013 onward[4]. Consequently the only solution suitable was to create static routes between Distribution, Core and Border for each IPv6 subnet used by the IPv6 Lab. This problem will slow down the overall IPv6 deploy because the OSPF protocol is the glue that keeps the network routing together, replacing its work by hand adding and removing static routes for network subnets is obviously discouraged and error prone.

The Internet Border routers use the BGP protocol to announce the CINECA IPv4 prefix towards the Internet, the same must happen for IPv6. For the IPv6 Lab purpose the *i02* router has been configured with a BGP peering to the GARR POP in Milan, meanwhile the *i01* router receives BGP

[3]September 2012

[4]The date was communicated to the CINECA network engineer by the CISCO customer support

updates from *i02* using the Internal BGP protocol. The activation of a second BGP peering from *i01* to the GARR POP in Bologna is scheduled for the middle of November.

4.3 Security

The study of IPv6 Security has been a fundamental part of my internship: we followed the bottom up strategy testing the IPv6 vulnerabilities from the Access layer to the Distribution one. Section 4.3.1 presents the IPv6 security most relevant exploits and countermeasures, meanwhile the rest of the Sections are about Data, Control and Management Plane security applied to each network layer.

4.3.1 Neighbor Discovery protocol vulnerabilities

As stated in Chapter 1 the Neighbor discovery protocol implements the IPv6 most important features using the following ICMPv6 packets[5]:

- Router Solicitation
- Router Advertisement
- Neighbor Solicitation
- Neighbor Advertisement
- Redirect

We have investigated those five ICMPv6 functionality in order to find vulnerabilities and estimate security risks for two categories of hosts:

servers - facing Internet traffic and hosting production environments like Web/application servers or critical services like DNS and NTP. Systems

[5] `http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xml`

engineers have complete control on those hosts, from the configuration to the management.

clients - Web/system clients like the employee's workstations, configured by systems engineers but managed completely by physical owners.

The rest of the section contains the security configuration details to protect the two host categories from IPv6 vulnerabilities.

Router Advertisement is a particular kind of ICMPv6 packet used to announce an IPv6 subnet prefix to a set of hosts in order to give them all the information needed to configure their network layer (see Section 1.5.5). An example is depicted in Figure 4.4 in which a router announces two different IPv6 subnets to its physicals ports. This workflow implies that hosts consider the Router Advertisement packet originated by a trusted router in the network: what happens if a compromised host exploits this trust and start sending forged announces? The answer is depicted in Figure 4.5: the forged Router Advertisement is propagated by the switch to all the hosts in the subnet forcing them to use a different default gateway instead of the real one[6].

This problem is a big concern for network engineers because once an attacker gains control to a host in a network it could create serious damages like packet forging, eavesdropping, man in the middle and denial of service attacks. Moreover an error in the network configuration of a host could lead to an overall network outage, so both internal and external threats are possible. There are two solution to this problem:

1. Secure Neighbor Discovery Protocol [3]: a trusted certificate authority issues a public/private key pair for each node of the network and the 64 bit host addresses part is generated applying an hashing algorithm like SHA-1 to a string obtained by a random number, the host's public key and the subnet prefix. The IPv6 address obtained is very hard to

[6]Operating systems like Linux and Windows set the last RA sender's address as default gateway accordingly with its priority (Low, Medium, High)

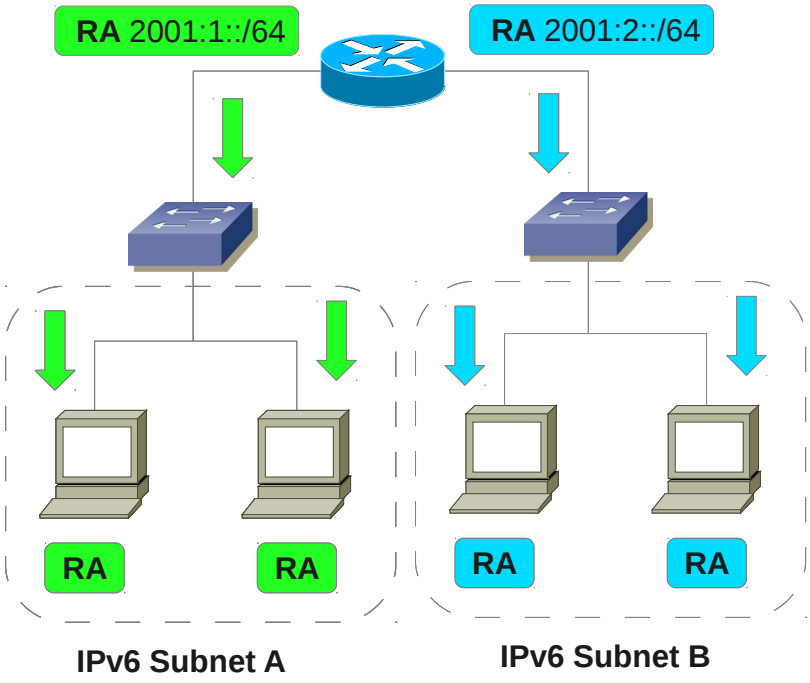

Figure 4.4: Example of the correct Router Advertisement data flow.

forge: the router's address becomes an assurance for the validity of the Router Advertisement.

2. Router Advertisement Guard [17]: an Access Control List for Layer 2 switches states the list of ports authorized to forward frames containing RA IPv6 packets[7].

The first solution is elegant and complete, it also prevents most of the attacks to the Neighbor Discovery Protocol, but the drawback is the need to provision a certification authority to issue public/private key pair for each node of the network. We decided to deploy the second solution because it

[7]It is sufficient to inspect the IPv6 header to find if the packet is a Router Advertisement.

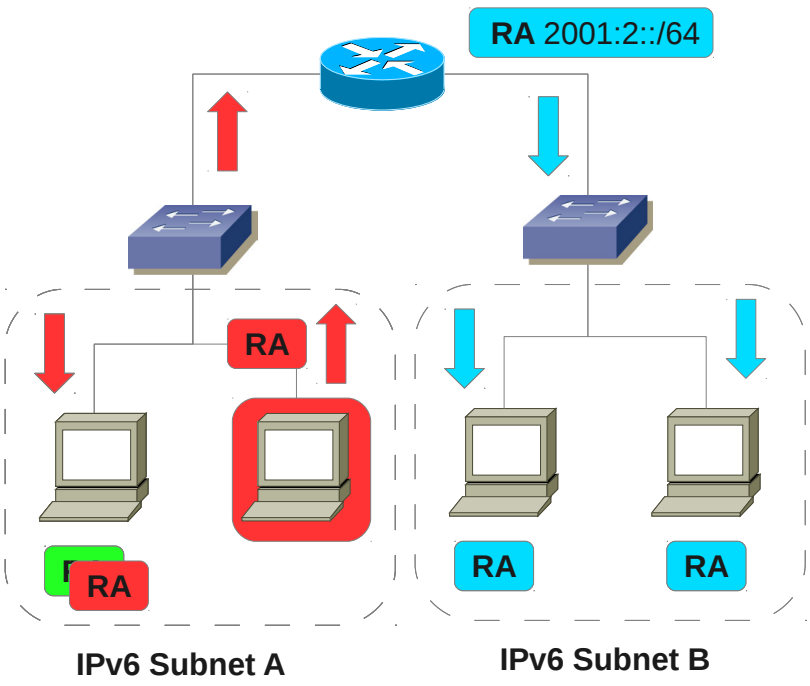

Figure 4.5: Example of a compromised host sending forged Router Advertisements.

offers a good trade off between management efforts and host troubleshooting, ensuring a strong level of security against RA poisoning/snooping without needing of additional system services. Moreover the SLAAC autoconfiguration is too dangerous for the *servers* category, because a malicious Router Advertisement or a technical error could potentially change the routing of an entire subnet, heavily affecting the services running on the hosts. The static configuration is not enough, autoconfiguration must be explicitly disabled. In Windows 2008 Server and Windows Seven it is necessary to execute the following command:

```
netsh interface ipv6 set interface ID routerdiscovery=disabled
```

where *interface ID* is the number assigned by the OS to the network card. The Debian GNU/Linux OS needs the following commands:

```
sysctl -w  net.ipv6.conf.default.autoconf=0
sysctl -w  net.ipv6.conf.default.accept_ra=0
sysctl -p
```

The ICMPv6 Redirect lets routers to inform hosts about route corrections to reduce their network latency. For example Router A could inform Host X that the fastest next hop available to reach Host Y is Router B instead of itself. Obviously the absence of authentication (in case Secure Neighbor Discovery is not deployed) means a malicious host may impersonate a router and change routes of another host in its subnet. An example is depicted in Figure 4.6: three hosts in a network joined by a switch (same IPv6 subnet and same VLAN id), one malicious host, one target host and another one unaware of the attack. The malicious host forges an ICMPv6 Echo Request packet using as sender the IPv6 address of the unaware host and as receiver the IPv6 address of the target one, then it sends the packet and waits for the ICMPv6 Echo Reply from the target host to the spoofed sender. The malicious host then could use the Echo Reply packet as payload for another forget packet, the ICMPv6 redirect one, impersonating the target host default gateway and forcing the target host to add a correction to the route for a specific host, in this case the unaware one. If the correction states the new gateway is the malicious host, then we have a Man in the Middle attack, otherwise it is a Denial of Service.
The Router Advertisement Guard forbids also the ICMPv6 Redirect forwarding by routers, but we are aware its deployment to all the network components will be slow due to:

- availability of router/switch firmwares supporting the RA guard;

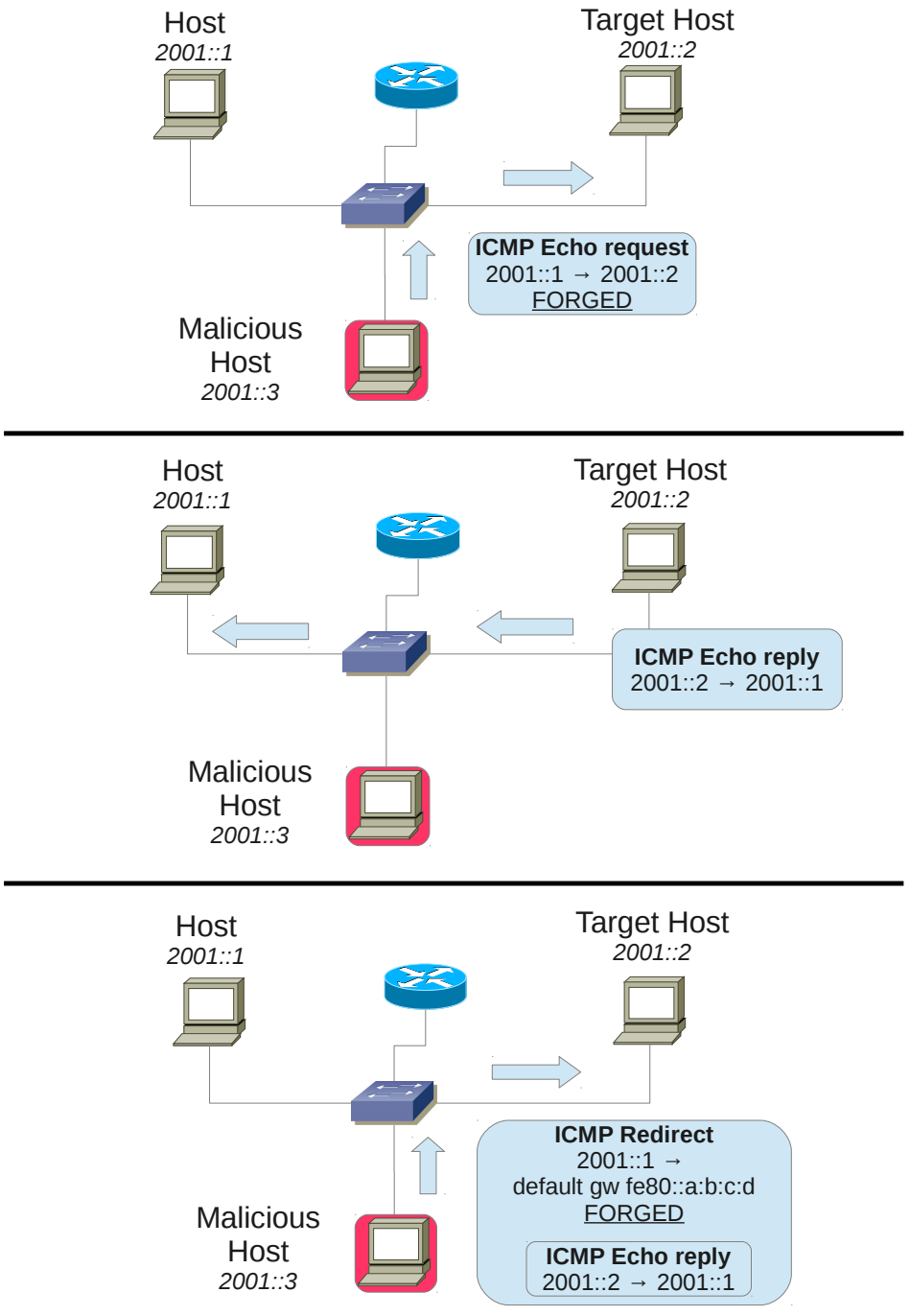

Figure 4.6: Example of a compromised host sending forged ICMPv6 Redirect packets. The attack is organized as follows: first the malicious host sends a forged ICMPv6 Echo Request to the target host using the spoofed IPv6 address of another host in the network as sender, then the target host will send an ICMPv6 Echo Reply to the spoofed host and finally the malicious host will send to the target one an ICMP Redirect using as evidence the ICMP Echo Reply to meet the Redirect requirements.

- provisioning and replacement of network components that will not support the RA guard in the future because network vendors will not supply the feature in their firmware updates.

Therefore the *servers* category must take additional precautions and avoid completely this kind of packet. The Windows hosts should simply deny the Redirect packets adding a rule to the firewall embedded in the OS, meanwhile the Debian GNU/Linux OS needs the following commands:

```
sysctl -w net.ipv6.conf.default.accept_redirects=0
sysctl -p
```

The *clients* category hosts are managed by end users so systems engineers have no control of this feature, meaning the Man in the Middle attack is possible among this kind of hosts. This is an acceptable risk for CINECA, the employees are a trusted group so they should not be tempted to harness their own working environment. Moreover it offers a good balance between the cost of the engineering effort to mantain security policies and the cost of an eventual a posteriori intervention after an attack.

The last feature analyzed is the Neighbor Solicitation and Advertisement: this is the IPv6 version of the ARP protocol, namely the possibility for hosts to discover which Layer 2 address (usually Ethernet MAC) is associated with an IPv6 one. This new feature is lighter than ARP because it uses multicast instead of broadcasts, but the principle is the same: a requestor asks for an address and another host will answer to it without requiring any authentication. The Man in the Middle attack based on the ARP cache poisoning in IPv4 has its natural counterpart in the Neighbor cache poisoning in IPv6. Moreover the Duplicate Address Detenction technique uses the Neighbor Solicitation to discover if the link local autoconfigured address is already used within the same broadcast domain: a malicious host could take advantage of this situation answering with a forged Neighbor Advertisement

each time it sees a Neighbor Solicitation flowing in its subnet denying the network access to other hosts.
CINECA considers this kind of attack an acceptable risk for both *servers* and *clients*, in the beginning it will be deployed only a tool called NDPmon [6] to monitor every IPv6 subnet of the CINECA network.

4.3.2 Data Plane protection

The Data Plane protection is the set of rules applied to filter the traffic between hosts to prevent spoofing attacks from malicious users.
The Access Control List chosen for the Distribution layer inbound traffic is the following:

- allow IPv6 packets with the routers' link local addresses as source,
- allow IPv6 packets with the routers' HRSP link local addresses as source,
- deny any Router Advertisement from the hosts,
- allow the IPv6 packets with source addresses **FE80::/64** (link local prefix) and destination address **FF02::/16** (multicast prefix),
- deny any ICMPv6 Redirect from the hosts,
- allow ICMPv6 traffic with source addresses **FE80::/64** and type Neighbor Discovery Advertisement or Solicitation,
- allow IPv6 packets originated from the IPv6 allowed subnet prefixes,
- allow IPv6 packets destined to IPv6 allowed subnet prefixes,
- deny the rest.

The Access Control List chosen for the Border layer inbound traffic (from Internet to the CINECA's network) is the following:

- deny IPv6 packets with the following IPv6 deprecated prefixes as source: **2002:e000::/20**, **2002:7f00::/24**, **2002:0000::/24**, **2002:ff00::/24**, **2002:0a00::/24**, **2002:ac10::/28**, **2002:c0a8::/32**, **2001:db8::/32**,
- allow ICMPv6 traffic with source addresses **FE80::/64** and type Neighbor Discovery Advertisement or Solicitation (allow communications between BGB peering routers),
- deny IPv6 packets with Routing Type 0 Extension header (deprecated by IETF because of they might be used to generate loop based denial of services to routers),
- deny IPv6 packets with source addresses **2001:760:2e0a::/48** (the CINECA's IPv6 prefix used for address spoofing)
- allow ICMPv6 type destination unreachable, echo request, echo reply, packet too big, parameter problem from the IPv6 addresses **2000::/3** (valid IPv6 allocated prefixes),
- allow TCP established connections' packets encapsulated in IPv6 ones with source **2000::/3** and destination **2001:760:2e0a::/49** (see Section 4.2.1),
- allow IPv6 packets with source **2000::/3** and destination **2001:760:2e0a:8000::/49** (see Section 4.2.1),
- deny IPv6 packets with source textbf2000::/3 and destination **ff05::/16** (deny inbound site scope multicast, it is only permitted inside the CINECA's network),
- allow IPv6 packets with source **2000::/3** and destination **ff00::/8** (allow global multicast for inbound traffic),
- deny the rest.

Finally the Core layer does not need Data Plane protection because it represents the CINECA network backbone and it must process packets as quickly

as possible, ACL would slow it down, it does implement only Control Plane protection.

4.3.3 Control Plane protection

The Control Plane Protection concerns the management of network traffic directed to routers like routing protocols (OSPF, EIGRP, etc...), management protocols (SNMP, ICMP, etc...) and remote access protocols (SSH, Telnet, etc...). This kind of traffic is not forwarded by routers but processed by their CPUs, so it could be a potential target for a denial of service attack by malicious hosts trying to saturate the network components' hardware. The best practice is to list all the protocol to manage, create ad hoc classes coupled with Access Control Lists and assign each protocol to a specific class. The usual role of a class is to rate limit the number of bytes per second going through the router's CPUs to specific threshold, stating which action perform if the threshold is met (for example drop or allow). The configuration values for the ACLs are strictly dependent from the routing hardware and from the average network load of the network. Let's go through an example for the OSPFv3 protocol (CISCO syntax):

```
class cppclass-ospf
  police cir 32000 bc 1000 be 1000
    conform-action transmit
    exceed-action transmit
    violate-action drop

class-map match-any cppclass-ospf-v3

ipv6 access-list cpp-ospf-v3
  permit 89 FE80::/10 any
```

The first part of the example is the definition of a traffic class called *cppclass-ospf*, set up to accept 32KB/s of network load with 1KB of temporary burst. The *conform-action* state the action to do in case the threshold is not met (in this case accept the incoming packet with the *transmit* keyword), the *exceed-action* states the action to do when the temporary burst threshold are met and the *violate-action* states the action to do when every threshold is exceeded (in this case drop the incoming packet with the *drop* keyword). The CINECA Control Plane protection overview is depicted in Figure 4.7, it applies to both IPv4 and IPv6 traffic but for the sake of clarity only the IPv6 relevant protocols are shown. There are six main traffic classes:

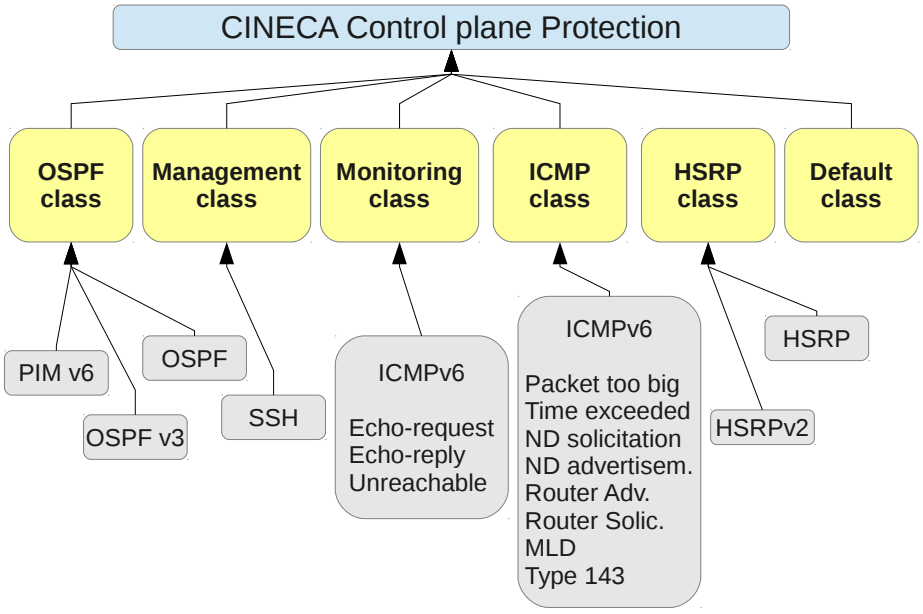

Figure 4.7: The CINECA's IPv6 Control Plane Protection for the actual routing hardware (October 2012).

OSPF - it contains the various flavour of the OSPF protocol and PIMv6.

Management - it contains the SSH protocol.

Monitoring - it contains the ICMPv6 packet types used by tools like *ping* to test the network availability and latency. Routers must answer to those packets because they are used by systems engineers and monitoring tools.

ICMP - it contains the ICMPv6 packet types related to IPv6 functionality like Neighbor Discovery. Routers communicate with each other using the IPv6 protocol, so they must be allowed to use its standard features like a standard IPv6 host does.

HSRP - it contains the HSRP protocol used by routers to elect a leader to act as default gateway for a network (see Section 2.2.4).

Default - all the traffic that does not match the above category is automatically added to the default class.

As stated above the configuration of the classes' thresholds is highly dependent from the routing hardware: the *d01*, *d02*, *c01*, *c02*, *i01*, *i02* have the same CPU capabilities from the Control Plane Protection point of view, therefore the same traffic classes are applied to the Distribution, Core and Border layers. The actual configuration details are not stated in this thesis for security purposes, moreover I believe they are not relevant for the scope of this thesis.

4.3.4 Management Plane protection

The Management Plane security has been copied from the best practices followed during the last years by CINECA for the IPv4 protocol, namely:

- restriction to the SSH protocol for secure remote login to network components (no Telnet or similar programs allowed),
- only one username is allowed to login to network components and users gain its privileges only after a login to a bastion host.

4.4 The Application layer

4.4.1 IPv6, Java and C

Nowadays Java and C are two of the most used programming languages in the world: the former is interpreted, static typed and focused on language abstractions, while the latter is compiled, static typed and focused on speed exploiting hardware optimization. The question is: how do they handle an IPv6 socket? How portable is the code written with the IPv4 only mindset? I tried to answer the above questions writing some code in both languages, testing a socket connection to hostname *www.google.com* port 80.

The keystone code is the one responsible to *translate* an hostname to its IP address using the DNS system: if the hostname is associated with both AAAA and A records, how the requestor sorts the IP addresses returned? For example, *www.google.com* is associated[8] with the IP addresses stated in Figure 4.8, in which we can count five IPv4 addresses and one IPv6 address. The operating systems offers usually a system call able to retrieve and sort DNS record results following some policies and rules; the Unix/Linux world uses a POSIX system call called *getaddrinfo* that implements RFC 3484 [12], the same happens for the Windows OS. The RFC states how select *source* and *destination* addresses for each IP packet generated by an host, using the IPv6 stack as much as possible. Figure 4.9 and Figure 4.10 shows Java and C code snippets that print all the IP addresses associated with *www.google.com* as sorted by the related syscall.

The Java Virtual Machine (JVM) adds an abstraction layer on top of the operating system one, sorting the IP addresses returned using additional policies. In particular two JVM options regulate this sorting:

-Djava.net.preferIPv4Stack default to *false*, it states if the JVM can use both IP stacks or if it must use only the IPv4 one;

-Djava.net.preferIPv6Addresses default to *false*, it states if IPv4 ad-

[8] October 2012

```
dig A www.google.com

;; QUESTION SECTION:
;www.google.com.  IN  A

;; ANSWER SECTION:
www.google.com.  IN  A  173.194.35.180
www.google.com.  IN  A  173.194.35.176
www.google.com.  IN  A  173.194.35.177
www.google.com.  IN  A  173.194.35.178
www.google.com.  IN  A  173.194.35.179

dig -t AAAA www.google.com

;; QUESTION SECTION:
;www.google.com.  IN  AAAA

;; ANSWER SECTION:
www.google.com.  IN  AAAA  2a00:1450:4016:801::1013
```

Figure 4.8: The Unix *dig* utility used to retrieve A records and AAAA records for the domain *www.google.com*

```
for (InetAddress address :
              InetAddress.getAllByName("www.google.com")){
        System.out.println(address.getHostAddress() + "\n");
}
```

Figure 4.9: Sample Java code correspondent to the Unix *getaddrinfo* function for the domain *www.google.com*

```
struct addrinfo *res;
char addrstr[100];
void *ptr;

getaddrinfo("www.google.com", NULL, NULL, &res);
while (res){
  inet_ntop(res->ai_family, res->ai_addr->sa_data,
        addrstr, 100);
  switch (res->ai_family){
    case AF_INET:
      ptr = &((struct sockaddr_in *)
                                res->ai_addr)->sin_addr;
      break;
    case AF_INET6:
      ptr = &((struct sockaddr_in6 *)
                                res->ai_addr)->sin6_addr;
      break;
  }
  inet_ntop (res->ai_family, ptr, addrstr, 100);
  printf ("IPv%d address: %s\n",
            res->ai_family == PF_INET6 ? 6 : 4, addrstr);
  res = res->ai_next;
}
```

Figure 4.10: Sample C code using the Unix *getaddrinfo* function for the domain *www.google.com* (some code has been omitted for code clearness).

dresses are preferred in DNS name resolving over the IPv6 ones or not.

This means the default JVM behaviour is to allow both IP stacks, but to prefer IPv4 addresses over IPv6 ones in DNS name resolving. On the contrary, the C code does not offers abstractions over the operating system syscalls, it uses the *getaddrinfo* function to resolve DNS names so the IP address are sorted only by the OS.

Now we have all the tools needed to open a socket in both Java and C, as Figure 4.11 and Figure 4.12. The Java *Socket* class constructor is used to set up and open a socket connection to an host port, it does not require strictly its IP address, it works also with hostnames. This means it embeds the functionality showed in Figure 4.9, and most notably it does not require to specify the IP protocol because it will be chosen transparently by the function itself. The C code is more verbose but it does the same thing, namely it uses the information returned by the *getaddrinfo* to fill the socket function parameters transparently to the rest of the code. This means applications written in both languages should run without any error on a dual stack IPv4/IPv6 host, as long as the developer follows the major networking coding guidelines while its writing the code. Conversely application written explicitly with the IPv4 mindset (for example opening a socket using explicitly the IPv4 protocol) may lead to networking issues during runtime.

4.4.2 IPv6 and Web/application servers

The transition from IPv4 to IPv6 for the Web begins from its major actors: the Web servers. CINECA systems engineers deploy web application on Apache Web server, Apache Tomcat and Red Hat JBoss: they have been tested within the IPv6 Lab to check their compliance with the new protocol and how well they handle requests in dual stack. The tests were designed to use particular features of the server applications:

Apache Web Server - Two Virtual Hosts configured to bind the TCP port 80 on a public IPv4 address and a global IPv6 one (both belonging to an

```
import java.io.*;
import java.net.*;

class SocketMain {
  public static void main(String[] args) {
    Socket javaSocket = null;
    String hostname = "www.google.com";
    try {
      javaSocket = new Socket(hostname,80);
    }
    catch (UnknownHostException e) {
      e.printStackTrace();
      System.exit(1);
    }
    catch (IOException e) {
      e.printStackTrace();
      System.exit(1);
    }
    javaSocket.close();
    System.exit(0);
  }
}
```

Figure 4.11: Sample Java code to create a Socket and connect to *www.google.com*

```
struct addrinfo *res, *aip;
int sock = -1;
int error;

getaddrinfo("www.google.com", NULL, NULL, &res);

for (aip = res; aip != NULL; aip = aip->ai_next) {
  sock = socket(aip->ai_family, aip->ai_socktype,
                aip->ai_protocol);
  if (sock == -1) {
    perror("socket");
    freeaddrinfo(res);
    return (-1);
  }

  if (connect(sock, aip->ai_addr, aip->ai_addrlen) == -1) {
    perror("connect");
    close(sock);
    sock = -1;
    continue;
  }
  break;
}

freeaddrinfo(res);
```

Figure 4.12: Sample C code snippet to create a socket and connect to *www.google.com* (some code has been omitted for code clearness, this example is taken from the Oracle Java IPv6 guide).

host's network interface). The IPv4 Virtual Host points to a directory with an *html* well formatted file showing the sentence *Welcome to IPv4!*, meanwhile the IPv6 Virtual Host uses the *mod_python*[9] module and points to a directory containing a Python script, that retrieves the *www.google.com* A record using IPv4 and write it to an html page together with the sentence *Welcome to IPv6!*. The deployment scheme is depicted in Figure 4.13.

Apache Tomcat - One Virtual Host configured to bind the TCP port 80 accepting both IPv4 and IPv6 traffic. A Java Spring[10] 3.0 Web Model-View-Controller Application is deployed in the Tomcat Servlet container, composed by:

- a JSP[11] presentation page,
- a Java class representing the Controller used to manage the View layer,
- a Java class that retrieves the DNS A record for *www.google.com* using the IPv4 stack.

The deployment scheme is depicted in Figure 4.14.

Red Hat JBoss - Same deployment as Tomcat, we did not test the JBoss EJB container.

The testing environment was composed by hosts equipped with the following software version:

- **OS** - GNU/Linux Debian 6.0 with Linux Kernel 2.6.32-5 x86_64
- **Apache** - Version 2.2.16 (Debian)
- **Tomcat** - Version 7 (Debian)

[9] `http://www.modpython.org/`
[10] `http://www.springsource.org`
[11] `http://www.oracle.com/technetwork/java/javaee/jsp/index.html`

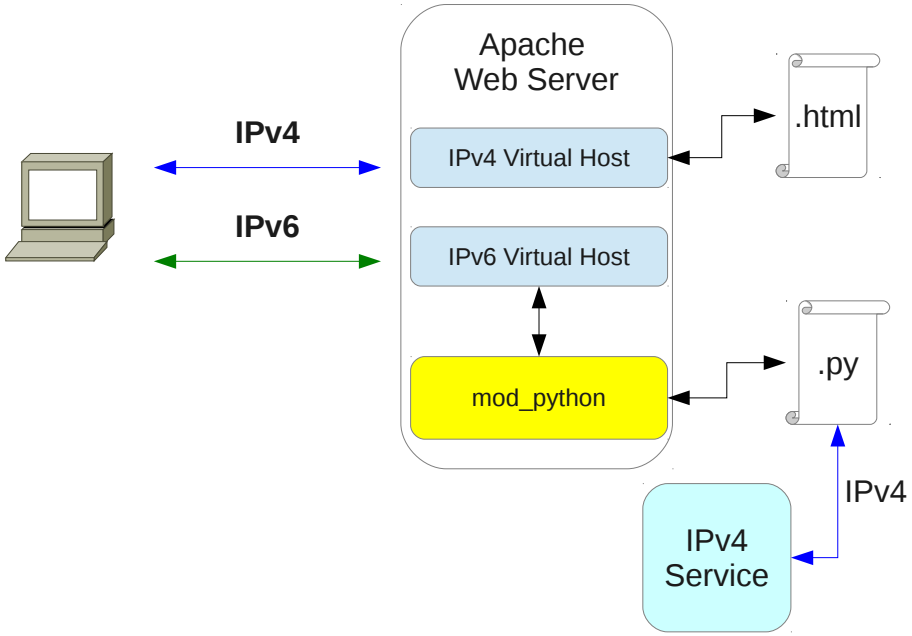

Figure 4.13: Deployment scenario to test the Apache web server with the IPv6 protocol.

- **JBoss** - Community Version 5/6/7 and Enterprise 5.1.2 (CINECA)

Tests have been executed deploying the Web/application servers on a server host (running Debian GNU/Linux) and making HTTP requests from two clients (one running Windows 7 and the other one running Debian GNU/Linux), both belonging to the IPv6 Lab but assigned to different IPv6 subnets. The correct behaviour of the applications has been established using the Google Chrome[12] on the client and monitoring the correct IP packet flow with the *tiptop* network utility on the Linux hosts (the server and one client). The Windows 2008 R2 OS has been tested only making HTTP requests from both clients to the running IIS 7 Web server[13] binded on TCP port 80 accepting traffic from every IPv4/IPv6 host. Results confirmed the expectations, the dual stack works fine on the operating systems tested.

[12] `https://www.google.com/intl/it/chrome/browser/`
[13] `http://www.iis.net/`

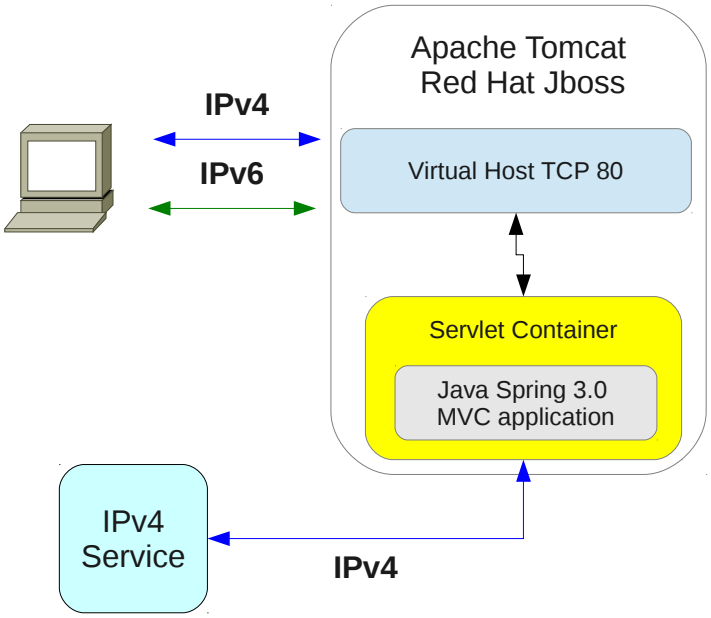

Figure 4.14: Deployment scenario to test the Tomcat web server / servlet container with the IPv6 protocol.

Chapter 5

Results and future work

I love it when a plan comes together!
Col. Hannibal Smith

The CINECA migration to IPv6 is an ongoing process that will take a lot of time to complete, this thesis represents the foundations on which all the subsequent work will stand. This chapter is a summary of the migration strategy: initial expectations, problems encountered, technical choices, results achieved and future work.

5.1 Starting steps

The CINECA Systems and Technologies Department (DSET) decided to start the process of supporting the IPv6 protocol in its infrastructure to gain experience in this new technology scenario. The motivations behind this choice were related to the recent IPv4 address exhaustion of the IANA main pool and the subsequent one of the APNIC and RIPE Regional Internet Registries, together with an evident effort by network vendors to support the IPv6 early adopters.
The initial questions and major concerns were the following:

- What are the differences between IPv4 and IPv6?

- Does the CINECA infrastructure need an upgrade to support IPv6?
- How will the web applications handle the new protocol?
- How should the infrastructure deal with IPv6 and IPv4?
- What migration strategy should CINECA choose for its infrastructure?

We decided to acquire IPv6 technical knowledge before answering any of the above questions. The documentation found on the Internet was often old and misleading, so we chose to stick to the following sources:

- CISCO technical books [15] [19]
- United States National Security Agency's technical documentation [1]
- SANS Institute technical documentation [2]
- GARR IPv6 webinars (web streaming from the GARR website)

The first big step was the requirement analysis that brought us to prefer the dual stack IPv4/IPv6 over other solutions like NAT64 and IPv6 to IPv4 tunnels. Subsequently we designed a working plan composed by macro steps, the major milestones for the migration, and then we broke them down into smaller steps obtaining the list stated in Section 3.3. The all process is described in detail in Chapter 3, meanwhile the description of the results obtained is stated in Section 5.2.

5.2 Results achieved

The IPv6 protocol has been investigated and a test case, the IPv6 Lab, has been deployed within the CINECA infrastructure. We shaped the Lab to simulate IPv6 Internet traffic between a set of server hosts and a set of

[1] http://www.nsa.gov/ia/_files/routers/C4-040R-02.pdf

[2] http://www.sans.org/reading_room/whitepapers/firewalls/cisco-router-hardening-step-by-step_794

client hosts, trying to make it as heterogeneous as possible to test all major software stacks supported by CINECA. In the process we managed to enable the IPv6 stack to the CINECA network skeleton, namely:

- the Distribution routers *d01* and *d01*;
- the Core routers *c01* and *c01*;
- the Border routers *i01* and *i01*.

We followed a bottom up deployment strategy: Access/Distribution layers first, then Core and finally Border joining the IPv6 Internet through the GARR network. For each layer we created ad hoc Management, Control and Data plane protection rules in order to secure the CINECA infrastructure in waterproof compartments before accepting IPv6 Internet traffic.
The CINECA software stacks have been tested splitting the work into two main parts:

- analysis of correct and reliable configurations to support IPv6 on the CINECA most used server applications, namely Apache Web server, Apache Tomcat and Red Hat JBoss. Creation of ad hoc Web applications to test the dual stack IPv4/IPv6 on the aforementioned software (see Section 4.4.2);
- creation of ad hoc socket programming code written in C and Java in order to discover what a developer should know to write IPv6 compliant software (see Section 4.4).

All the results obtained are described in detail in Chapter 4.

5.3 Issues encountered

The first issue encountered was the lack of IPv6 support by CISCO routers firmware, that on paper should have supported the IPv6 protocol, but in practice the implementation was really poor and rudimentary. Right after the

IPv6 Lab deployment we had many difficulties to make a simple *ping* between two IPv6 subnets, obtaining strange multicast traffic anomalies. We decided to upgrade the firmware of the *c01* and *c02* Core routers, then *d01* and *d02* Distribution routers and finally *i01* and *i02* Border routers. The Distribution layer upgrade required a lot of planning and engineering work because most of the production environments and employees workstations are connected in high availability to those routers: a takeover has been planned carefully and the upgrade went fine. The big obstacle we encountered was a missing feature in the latest CISCO firmware release, namely the compatibility between Virtual Router and Forwarding (VRF) and the OSPFv3 protocol, described in detail in Section 4.2.6. CISCO will fix the issue for 6509 routers in the first quarter of 2013, so the only possible solution was to create static routes in the Distribution and Core layers to manage the IPv6 Lab subnets traffic. Network vendors are currently promoting IPv6 announcing a full IPv6 support in their network products, but there is still a lot of work to do in our opinion.

The security analysis (described in detail in Section 4.3) brought our attention to security features like the Router Advertisement Guard, so we investigated the vendor support for the actual routers/switches firmwares. We found compatibility problems in some layer 2 switches dedicated to the employees workstations, and most surprisingly the same issue in the virtual switches implemented in the VMWare software running onto the CISCO UCS (see 2.2.8). The DSET started the process to provision new hardware switches for the next months, and the VMWare support has been contacted to request the RA guard feature in its virtualization software.

Another issue, if we call it that way, was approaching IPv6 with the IPv4 mentality, trying to apply the well consolidated best practices to the new protocol without any changes. Address autoconfiguration, DHCPv6, multiple addresses on a single network interface, etc. . . are new and valuable features that should be used in the new IPv6 mentality.

All the issues encountered and their technical solutions are described in detail in Chapter 4.

5.4 The next steps

Two infrastructure components are still needed before accepting any internal customer for IPv6 production services: the firmware upgrades to enable the OSPFv3 routing within the VRF instances and the upgrade of the automation software used by the CINECA systems engineers for DNS zones handling. The former step is blocked by CISCO software development, meanwhile the latter is a missed goal of this thesis due to the heavy workload required for the other steps. As stated in Section 4.2.3, only one DNS zone supporting IPv6 has been created and its purpose was only to support the IPv6 Lab. The GARR Network Operation Centre (NOC) has already granted the delegation for the reverse DNS lookup to CINECA, and a reverse zone file for the **2001:760:2e0a/48** prefix has been filled with some PTR records manually to test its correct behaviour. CINECA systems engineers uses automation software to add various kind of DNS records to its zones, adding the support for the AAAA record is one major step of the next months.

Once the above steps are completed the plan is to proceed with what stated in Section 3.3, accepting the first CINECA internal customer to host a production service using the IPv4/IPv6 stack.

Conclusions

This thesis represents the initial steps of a long process, the migration of the CINECA infrastructure to the dual stack IPv4/IPv6. I worked five months with CINECA systems and network engineers to find a suitable strategy for the company, shaping the work around the software stacks currently running at CINECA.

Several achievements have been reached during those months:

- the CINECA network skeleton has been updated to support the IPv6 protocol;
- new Management/Control/Data Plane protection security policies have been discovered to protect the CINECA network;
- the major software stacks (operating systems, Web server applications, etc...) have been tested with IPv6, and a list of ad hoc configuration has been written to facilitate the forthcoming deployments of IPv6 aware software;
- a complete and detailed work plan to carry on the migration in the next months has been created and reviewed by the DSET department.

The first CINECA internal customer for an IPv6 production service will be ready in Q1/Q2 2013, but the realistic time to market will depend heavily from other factors like the availability of CISCO router firmwares supporting both VRF and OSPv3, the upgrade of the automation software related to the managing of critical systems like the DNS and finally the time required for the CINECA systems engineers IPv6 technical training.

Appendix A

Network Diagram Icons

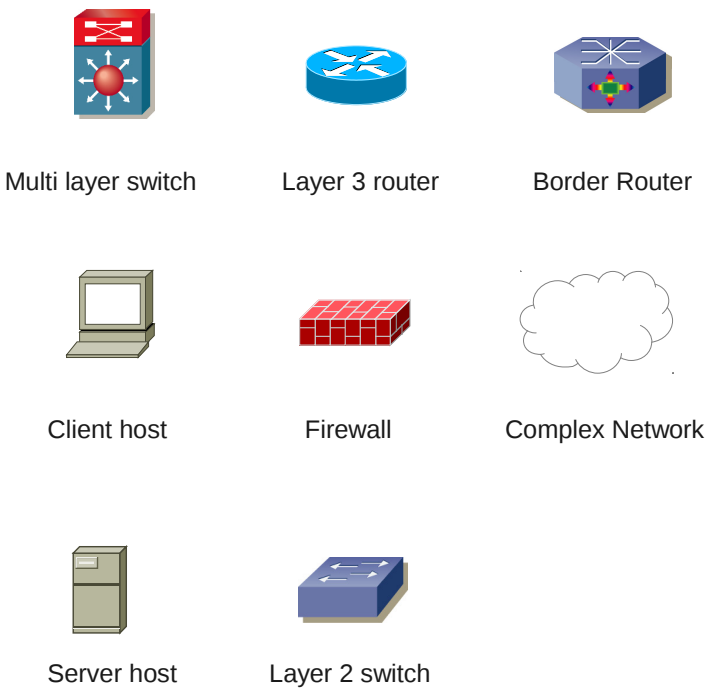

Figure A.1: Network icons used in this thesis.

Bibliography

[1] Internet Protocol. RFC 791, RFC Editor, September 1981.

[2] *IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks: Technical and editorial corrections*. IEEE (std.). IEEE, 2001.

[3] Arkko, J., Kempf, J., Zill, B., and Nikander, P. SEcure Neighbor Discovery (SEND). RFC 3971, RFC Editor, March 2005.

[4] Bagnulo, M., Matthews, P., and van Beijnum, I. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 2675, RFC Editor, August 1999.

[5] Bagnulo, M., Sullivan, A., Matthews, P., and van Beijnum, I. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147, RFC Editor, April 2011.

[6] Beck, F., Cholez, T., Festor, O., and Chrisment, I. Monitoring the Neighbor Discovery Protocol. In *The Second International Workshop on IPv6 Today - Technology and Deployment - IPv6TD 2007* (Guadeloupe/French Caribbean, Guadeloupe, Mar. 2007).

[7] Borman, D., Deering, S., and Hinden, R. IPv6 Jumbograms. RFC 2675, RFC Editor, August 1999.

[8] Carpenter, B., and Moore, K. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056, RFC Editor, February 2001.

[9] Cerf, V. G., and Khan, R. E. A protocol for packet network intercommunication. *IEEE TRANSACTIONS ON COMMUNICATIONS 22* (1974), 637–648.

[10] Conta, A., Deering, S., and Gupta, M. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443, RFC Editor, March 2006.

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

[12] Draves, R. Default Address Selection for Internet Protocol version 6 (IPv6). RFC 3484, RFC Editor, February 2003.

[13] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and Carney, M. Dynamic Host Configuration Protocol for IPv6 (DHCPv6. RFC 3315, RFC Editor, July 2003.

[14] Fuller, V., and Li, T. Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. RFC 4632, RFC Editor, August 2006.

[15] Hogg, S., and Vyncke, E. *IPv6 Security*, 2st ed. Cisco Press, 2011.

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

[17] Levy-Abegnoli, E., de Velde, G. V., Popoviciu, C., and Mohacsi, J. IPv6 Router Advertisement Guard. RFC 6105, RFC Editor, February 2011.

[18] Li, X., Bao, C., and Baker, F. IP/ICMP Translation Algorithm. RFC 6145, RFC Editor, April 2011.

[19] McFarland, S., Sambi, M., Sharma, N., and Hooda, S. *IPv6 for Enterprise Networks*. Networking Technology Series. Cisco Press, 2011.

[20] Mockapetris, P. Domain Names - Implementation and specification. RFC 1035, RFC Editor, November 1987.

[21] Narten, T., and Draves, R. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 3041, RFC Editor, January 2001.

[22] Narten, T., Nordmark, E., Simpson, W., and Soliman, H. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, RFC Editor, September 2007.

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

[24] Plummer, D. C. An Ethernet Address Resolution Protocol. RFC 826, RFC Editor, November 1982.

[25] Postel, J. Internet Control Message Protocol. RFC 792, RFC Editor, September 1981.

[26] Rekhter, Y., Li, T., and Hares, S. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, January 2006.

[27] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and Lear, E. Address Allocation for Private Internets. RFC 1918, RFC Editor, February 1996.

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

Acknowledgements

The first thank you goes to all the people I met during the time spent in Ireland, in particular the Amazon crew: it has been an amazing experience, one of the best of my life.
The rest will be in italian, I am sorry for the english readers!

Sette anni di studio, dal 2005 al 2012, quasi non mi sembra vero di aver finito. Si chiude un capitolo molto bello della mia vita, e molto probabilmente chi sta leggendo ora è una delle persone che hanno contribuito a rendere questo viaggio indimenticabile. Devo ringraziare tanti amici, ora sparsi in giro per l'Italia e per il mondo, non voglio nominarli tutti altrimenti correrei il rischio di dimenticare qualcuno. Grazie Grazie Grazie, mi sento davvero fortunato ad avere tutti voi nella mia vita.

Per quanto riguarda questa tesi, devo ringraziare tutto il DSET del CINECA per la pazienza e il supporto tecnico, in particolare Denis, Mauro e Alessandro che mi hanno seguito passo a passo insegnandomi i rudimenti del loro difficile mestiere, Claudio e Stefano per aver reso l'atmosfera in ufficio allegra e piacevole.

A Gabriele D'Angelo e Moreno Marzolla va un ringraziamento speciale per quanto mi hanno insegnato negli ultimi mesi della mia carriera universitaria, non potrò mai smettere di dire loro grazie.

Alla mia famiglia vanno gli ultimi, ma non meno importanti, ringraziamenti: grazie per aver reso possibile questa esperienza, sia finanziariamente sia supportando ogni mia scelta di vita.

License