34352
ADVANCED PROTOCOLS
SPRING 2006

FINAL REPORT

Internet Protocol version 6 and Transition Techniques

Piotr Szymański (s053750)
szyman@magres.net

Abstract

This report introduces the next version of the Internet Protocol (IPv6) and gives an overview of its features. A description of the addressing scheme, packet structure, and transition mechanisms for the protocol, along with other issues, is also given. Finally, a short discussion on the protocol features and associated problems, including a summary of the topic, is presented.

Table of contents

1. Introduction

In this section I will describe why this is a topic of interest to me, and give an overview of the issues and features related to IPv6.

1.1. Motivation for the choice of topic

The reason why I have decided to pursue this topic is twofold.

First, as I am interested in various networking concepts, it is important for me to be familiar with a technology that might, in the near future, become the basis for worldwide communication. The fourth version of the Internet Protocol is already being embedded in all kinds of devices to enable them to communicate in a standard way, but with the advent of IPv6 this trend might become even more widespread.

The second reason is the fact that I am an operator of a small network which provides hosting services to customers. The shortage of IPv4 address space is an issue I have experienced myself, as it is quite difficult to obtain new addresses that are often a must for providing some services. That is why I hope that the new version of the Internet Protocol will be a solution to this problem, and that the public global IP address will no longer be considered a precious resource as it is today.

1.2. The need for a new Internet Protocol

The fourth version of the Internet Protocol does its job very well. It is used on virtually all computers and devices connected to the Internet. There is just one significant problem that prevents its further adoption across the world – a problem that resulted from the protocol's unforeseen popularity – the shortage of address space.

As IPv4 addresses are 32-bit numbers [1], there is a limit of 4,294,967,296 possible unique addresses. Although this seems to be a very large number, the addresses were not assigned in an optimum manner, leading to the situation where a huge number of addresses was assigned but not used. Furthermore, it was not possible to assign all the available addresses to actual use, as a part of the address space had to be reserved for special purposes. But the single most important reason for the current lack of new addresses was the classful assignment approach, which is now difficult to reverse as it would require taking back addresses from companies and organizations to whom they were assigned.

Apart from the shortage of the address space, IPv4 suffered from numerous small problems that had surfaced as the protocol became more and more popular [2]. However, none of these problems were important enough to be the reason for shifting to a new IP version. Most of them could be resolved through modifications to the existing protocol.

The *Internet Engineering Task Force*, being aware of the problems of IPv4, had two choices in providing a satisfactory solution [3.a]. A completely new internetworking protocol could be introduced, that would address all the issues with IPv4. But this solution seemed to pose problems, as IP was already a well-tested and widely-adopted protocol. Instead, it was decided that a new version of the IP protocol would be developed.

1.3. Overview of changes in IPv6

The new version of the Internet Protocol, called Internet Protocol version 6 (IPv6) or Internet Protocol Next Generation (IPng), was to address problems that were found in the previous, or current version. As IPv6 is a new version of an existing protocol it should be analyzed and considered in this way – through the changes that were introduced as compared with the previous version. The basic functionality, motivation and the reasons for the protocol remained the same. A list of the most important changes is presented below [3]:

- **Larger address space**
  To eliminate shortage of addresses and the need for temporary solutions like NAT.
- **Hierarchical addressing**
  Each IPv6 address has an embedded hierarchy which makes routing packets across the network easier and faster by minimizing the size of the routing tables.
- **Simpler autoconfiguration**
  Enable hosts to dynamically discover the surrounding network even without such services as DHCP.
- **Better support for multicasting**
  This was not a popular or widely-used feature in IPv4, as even though theoretically present, many routers did not support it. [3]
- **Added anycast – a new type of address**
  A new type of address that enables a single host from a certain group to be reached. [4]
- **Simplified packet header format**
  The header of the main IPv6 packet was simplified to reduce the processing time required at the router for the common-case packet.
- **Improved support for extensions and future options**
  All fields that were not commonly used in the IPv4 packet header were moved to additional extension headers. Furthermore, a provision for future improvements was achieved in this way.
- **Quality of service support**
  The IPv6 packet header contains fields that will enable the classification of packets into traffic flows and traffic classes.
- **Built-in authentication and privacy capabilities**
  The IP security (IPsec) standard is an integral part of the IPv6 specification.
- **Large („jumbo") packets support**
  This feature is intended for networks with a large Maximum Transmission Unit (MTU) and systems that can process information very fast (e.g. supercomputers). Packets whose size exceeds the 64-kilobyte boundary can be transmitted in this way.

1.4. IPv6 timeline

Even though IPv6 is still not widely used as of 2006, it is not a new protocol. Initial work began in the **early 1990s**, when the Internet Engineering Task Force began investigating the problem of IPv4 address shortage [5]. In **December 1993**, RFC 1550 was published which issued a call for proposals about the next generation Internet Protocol [6]. In **October 1994**, an Internet Draft describing the *Next Generation of IP* recommended by the *IPng Area Directors of the Internet Engineering Task Force* was published [7], indicating the adoption of the protocol by the IETF. Work then continued, and in **December 1995** the first Request for Comments document specifying the protocol was made available – RFC 1883: "*Internet Protocol, Version 6 (IPv6) Specification*" [8]. The newest version of that document is now RFC 2460, which was published in **December 1998** [4].

But as we can all see, IPv6 is not anywhere near global adoption. The reason for this are the numerous problems with the transition from IPv4 to IPv6. The most important ones are overviewed in the following section.

1.5. Issues with transition

The benefits of IPv6 are quite clear, why is it then that every computer in the world is not already running this protocol? Even though IPv6 was designed to be the new version of IPv4, the transition from one protocol to the other is not as smooth as one might imagine. The reasons for this are listed below:

1.5.1. IPv6 is not compatible with IPv4

Although these are both versions of the same protocol, IPv6 is not backwards compatible with IPv4. That means that IPv6 hosts cannot directly communicate with IPv4 hosts, and vice versa. This in turn leads to the formation of two global networks – the IPv4 Internet, and the IPv6 Internet, which cannot communicate with each other.

1.5.2. Transition cannot happen "overnight"

One might imagine, that everyone could just switch to IPv6 at the same moment. Unfortunately, this is not possible in practice due to enormous problems with coordination. The transition from IPv4 to IPv6 has to be a gradual process, with both protocols coexisting for a very long time.

1.5.3. Software and hardware compatibility

To make the transition a reality, the network infrastructure (routers, routing algorithms, protocol stacks, etc.) has to be compatible with the new protocol. This actually represents the simpler part of the changes that need to be introduced [9], because most of that infrastructure is managed by a limited number of specialized companies, such as backbone operators and Internet Service Providers. The other part are user applications and higher-level protocols, which need to accommodate the new address format (Layer 4 protocols which need to be modified are limited in number, but there exists a large amount of Layer 7 protocols that embed IP addresses within their payload, and this means that they will have to be updated in order to operate with IPv6).

1.5.4. NAT and CIDR slowed down adoption

The primary problem with IPv4 is the shortage of address space. As this problem was already foreseen in the early 1990s [5], and no alternative to IPv4 was readily available, other measures were undertaken to delay the moment when no new addresses could be assigned.

One of them was CIDR – the Classless Inter-Domain Routing strategy [11] which was a proposal to do away with the *classful* assignment scheme and instead introduce a much more address-efficient solution. Indeed, one of the problems that CIDR was to address is:

> "*Exhaustion of the class B network address space. One fundamental cause of this problem is the lack of a network class of a size which is appropriate for mid-sized organization; class C, with a maximum of 254 host addresses, is too small, while class B, which allows up to 65534 addresses, is too large for most organizations.*" [11]

Another widely adopted solution was NAT – Network Address Translation [10], which enabled computers having IP addresses that were not globally unique (so-called *private* addresses) to communicate with hosts on the global Internet. NAT made it possible to connect many computers to the Internet with just a single global public IP address, therefore reducing the need for such addresses.

When both of these solutions were applied, the problem of the IPv4 space shortage, which was the major driving force behind the adoption of IPv6, was diminished and the incentive for companies to invest in IPv6 infrastructure became weaker.

1.5.5. Cost

One of the more important factors that hinders the shift to IPv6 are the costs required to upgrade the perform the transition. According to [12], the cost of shifting the U.S. federal government to IPv6 is estimated at around $75 billion. The costs for organizations and companies could also be quite high, and without clear reasons (see the previous section on NAT and CIDR) they might be reluctant to do so.

1.5.6. "Chicken and egg" problem

Daniel J. Bernstein in his discussion on IPv6 touches upon the problem of "who should switch first":

> *Before clients can be safely deployed on public IPv6 addresses, practically every server will have to learn how to talk to those clients. Before servers can be safely deployed on public IPv6 addresses, practically every client will have to learn how to talk to those servers.* [9]

This illustrates one of the problems that arise due to a lack of backward compatibility with the existing protocol. There are no incentives for the users to switch to IPv6 as long as there are no available servers providing popular services, such as Google, CNN, Wikipedia, etc., that they know and use in the IPv4 network. On the other hand, there is no incentive for the servers to switch to IPv6 if there are no users who could interact with their services.

2. Addressing scheme

In this section I will give an in-depth description of the addressing scheme used in IPv6.

2.1. Address size and address space

An IPv6 address is a 128-bit number. This means that there are 2^{128}, or 3.4×10^{38} unique addresses. In order to realize the size of the address space, one can compare it with the estimated age of the Universe, which is 13.7×10^{9} years, or 4.3×10^{17} seconds [13]. That is, for each second which has elapsed since the Big Bang, one could assign about sextillion (10^{21}) addresses.

The address space has been made so large for two reasons. First of all, to make sure that an address shortage will never ever occur again. Still, a much smaller number would be quite sufficient for this purpose. The other reason is the hierarchical allocation model, in which the addresses will be assigned to users in large continuous blocks. Such an approach will allow for simplified routing, as the size of the routing tables will decrease significantly. This issue will be discussed later on.

2.2. Text representation of addresses

Owing to the fact that IPv6 addresses are so long, a new notation scheme had to be devised, as the IPv4's dotted-decimal notation [15] would be too cumbersome. An example IPv6 address in dotted-decimal notation would look like this:

```
1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16
```

2.2.1. Hexadecimal notation

A new hexadecimal notation scheme [14] for IPv6 addresses was introduced:

> *The preferred form is x:x:x:x:x:x:x:x, where the 'x's are one to four hexadecimal digits of the eight 16-bit pieces of the address.* [14]

In this notation the 128-bit address is divided into 16-bit groups, with each group being represented as hexadecimal digits. The group themselves are separated using the colon character. An example IPv6 address in this notation would be:

```
ABCD:EF01:0000:0000:6789:EF01:0345:6789
```

To make the text representation shorter, one can remove leading zeros in each group. Note that at least one digit must remain in a group. The above address after this transformation would look like this:

```
ABCD:EF01:0:0:6789:EF01:345:6789
```

To reduce the size of the address even further, one can apply so-called *zero compression*, which allows a single consecutive occurrence of groups containing zeros to be replaced by a double colon (::). After applying *zero compression* to the above address, it will have the following form:

```
ABCD:EF01::6789:EF01:345:6789
```

It is important to note that *zero compression* can be used only once within a single address, i.e. an address of the form below is invalid, as it provides for ambiguous expansions:

```
ABCD:EF01::4567::6789
```

2.2.2. Mixed notation

As IPv6 is expected to coexist with existing IPv4 infrastructure for a long time, a special notation was introduced for such mixed environments. In this notation, the last four bytes of an IPv6 address are represented in dotted-decimal notation, while the rest of the address is written in hexadecimal notation. A sample address in this form is presented below:

```
0:0:0:0:0:FFFF:129.144.52.38
```

2.2.3. Prefix representation

An address prefix is used to represent a group of addresses having a certain number of left-most (i.e. most significant) bits that are equal. This number of bits is called the *prefix length*.

Address prefix representation is done in a similar way as in the case of IPv4 prefixes. A representation of a prefix has the following format:

```
ipv6-address/prefix-length
```

where *ipv6-address* is one of the address forms listed in the previous sections and *prefix-length* is a decimal number *specifying how many of the leftmost contiguous bits of the address comprise the prefix* [14]. An example of an IPv6 prefix representation is given below.

```
ABCD:EF01:0000:0000:6789:EF01:0000:0000/96
```

2.2.4. Base-85 notation

RFC 1924 [20] proposes another alternative to the problem of representing IPv6 addresses in text. It claims that the variable-length nature of the compressed hexadecimal notation described above is "*ugly*" and "*awkward*", and thus suggests a fixed-length (20 bytes) notation that involves converting the address to a Base-85 form using the following alphabet:

```
'0'..'9', 'A'..'Z', 'a'..'z', '!', '#', '$', '%', '&',
'(', ')', '*', '+', '-', ';', '<', '=', '>', '?', '@',
'^', '_', '`', '{', '|', '}', and '~'.
```

An example address in this form would be:

```
4)+k&C#VzJ4br>0wv%Yp
```

This notation did not find any wider practical usage, but is interesting due to the fact that it nicely illustrates the problem of representing IPv6 addresses in written form.

2.3. Types of addresses

IPv6 addresses are used to identify individual network interfaces and sets of interfaces [14]. The protocol defines three main types of addresses. Note that there is no *broadcast* address type in IPv6 – its function has been replaced by the *multicast* address.

2.3.1. Unicast type

A unicast address is an identifier for a single network interface. A packet that is sent to a unicast address is delivered to the unique interface identified by this address. Unicast addresses are the most commonly used types of addresses in the network.

2.3.1.1. Original model

The original model of the *Global Unicast Address Format*, as described in RFC 2374 [16], proposed that the unicast address should have a very strict internal hierarchy embedded inside the address itself. This hierarchy was meant to reflect the physical structure of the network and thus help with packet routing, as some of the information required for proper routing would be already embedded inside the address.

3	13	8	24	16	64
FP	TLA	RES	NLA	SLA	Interface ID

Figure 1: Global Aggretable Unicast Address format

The format of the *Global Aggretable Unicast Address* is shown above. The first three bits of the address were the Format Prefix (FP), a constant value of binary 001, that would identify this type of address. Then, a series of Aggregation Identifiers followed:

- Top-Level Aggregation Identifier (TLA) – a 13 bit number used for identifying the top level in the routing hierarchy, e.g. backbone networks,
- Next-Level Aggregation Identifier (NLA) – a 24 bit number that would allow an ISP organization that was assigned a TLA to structure its network,
- Site-Level Aggregation Identifier (SLA) – a 16 bit number used by the end-user organization to create its own subnet addressing.

The remaining parts of the address are the 8-bit reserved field (RES), and the 64-bit interface identifier.

Due to the fact that IPv6 addresses are so long, it is possible to use hardware addresses of the network interface cards used on a given link as the Interface IDs. The protocol specification [16] defines the Interface ID to be constructed in IEEE EUI-64 format [18]. In the case of Ethernet, this simply means that the Interface ID is constructed using the MAC address of the network adapter.

2.3.1.2. Current model

The model described above, as proposed by RFC 2374, has been made obsolete and replaced by a specification defined in RFC 3587 [17], which is the current one, as of writing this document. The reason for replacing that scheme was a concern that it was not the best technical solution at this stage of IPv6 development, and that the proposed structure might be too strict and more allocation freedom for Internet Registries is needed [3.b].

n bits	m bits	128-n-m bits
global routing prefix	subnet ID	Interface ID

Figure 2: Global Unicast Address format

A new address format was thus introduced, having the structure as shown above. The *global routing prefix* is *a value assigned to a site (a cluster of subnets/links)* [17], and is intended to be structured hierarchically as needed by the Regional Internet Registries (RIRs) and Internet Service Providers (ISPs). The subnet ID is an identifier within a site and can also be structured as needed by site administrators.

The specification [17] requires that all unicast addresses, except for those whose *global routing prefix* starts with a binary value of 000, are to have an Interface ID that is 64 bits in length. In other words, the combined length of the *global routing prefix* and the *subnet ID* must be 64 bits.

2.3.2. Anycast type

An anycast address is an identifier for a set of interfaces, i.e. it is assigned to more than one network interface. When a packet is sent to an anycast address, it is delivered to the *nearest* interface having this address. The definition of *nearest* is according to a routing protocols' measure of distance. [14]

Anycast addresses are allocated from the unicast address space and are therefore indistinguishable from unicast addresses.

The expected use of this type of address is to identify a set of routers belonging to an organization providing Internet service. One could imagine that a certain router pool would be identified by a single anycast address – in this way load-balancing could be achieved.

n bits	128-n bits
subnet prefix	0000000000000000000000

Figure 3: Subnet-Router address format

Another example is the *Subnet-Router* address, defined in [14], which identifies a set of routers on a given subnet. The format of this address is shown above. Such an address plays an important role in the autoconfiguration feature of IPv6 as it enables a host to automatically find a gateway router on its subnet. Note that in the case of IPv4, the "default gateway" has to be manually configured (or by using some other means of autoconfiguration, such as DHCP) for the host to be able to communicate with the external network.

2.3.3. Multicast type

A multicast address is an identifier for a set of interfaces. A packet that is sent to a multicast address is delivered to all interfaces having this address. [14]

2.3.3.1. Address format

A multicast address in IPv6 has a fixed format:

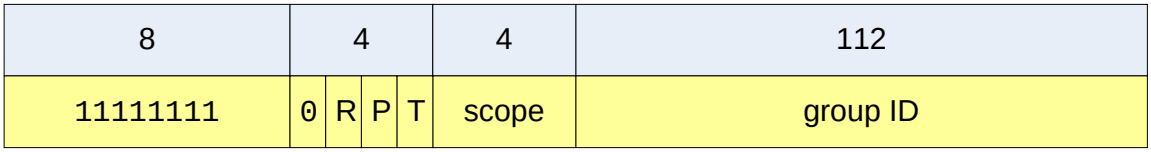

Figure 4: Multicast address format

It begins with eight binary one's, followed by four binary flags:

- The first flag is reserved and must be set to 0.
- The second and third flags, R and P, are used for Rendezvous Point embedding and Unicast-Prefix-based IPv6 addresses, respectively. [14]
- The fourth flag, T, indicates whether this is a permanently-assigned ("well-known") by the Internet Assigned Numbers Authority (IANA), or a non-permanently-assigned ("dynamically" assigned) multicast address.

The *scope* field is *used to limit the scope of the multicast group* [14]. The following values (given in hexadecimal format) are meaningful, other values are marked either as reserved or unassigned:

1	Interface-Local	5	Site-Local
2	Link-Local	8	Organization-Local
4	Admin-Local	E	Global

The meaning of the above scope types is described below:

- **Interface-Local** – the group is valid only on the local interface, i.e. is used for loopback transmissions to the node,
- **Link-Local** – the group spans a single link,
- **Admin-Local** – the group is defined manually by administrators,
- **Site-Local** – the group is intended to span a single site,
- **Organization-Local** – a collection of sites within a single organization,
- **Global** – there is no limit to this scope, i.e. this scope is equivalent to the whole Internet.

The *group ID* field identifies the multicast group. A permanently-assigned group ID retains its meaning irrelevant of the scope. Following the example given in [14], if a group of NTP servers is assigned a permanent multicast address group ID of (hexadecimal) 101, then:

> *FF01:0:0:0:0:0:0:101 means all NTP servers on the same interface (i.e., the same node) as the sender.*
> *FF02:0:0:0:0:0:0:101 means all NTP servers on the same link as the sender.*
> *FF05:0:0:0:0:0:0:101 means all NTP servers in the same site as the sender.*
> *FF0E:0:0:0:0:0:0:101 means all NTP servers in the Internet.*

On the other hand, the non-permanently-assigned multicast addresses are meaningful only within their scope, that is the same group ID could have a different meaning within a different scope.

2.3.3.2. *Pre-defined multicast addresses*

Some well-known multicast addresses are predefined and are required for the network to function properly. Note that these addresses are valid only for the explicit scope they have been defined for.

2.3.3.2.1. Reserved addresses

Addresses of the following form are reserved and cannot be used:

```
FF0X:0:0:0:0:0:0:0
```

where **X** ranges from 0 to F.

2.3.3.2.2. All Nodes addresses

The following addresses identify all IPv6 nodes within the local interface (scope 1) and the local link (scope 2), respectively:

```
FF01:0:0:0:0:0:0:1
FF02:0:0:0:0:0:0:1
```

Note that the second address is the functional equivalent of the IPv4 *broadcast* address type, as it allows all nodes on the local link to be reached. The first address can be considered as a *broadcast* address for the loopback interface.

2.3.3.2.3. All Routers addresses

The following addresses identify a group of all IPv6 routers on the local interface (scope 1), the local link (scope 2) and the local site (scope 5), respectively:

```
FF01:0:0:0:0:0:0:2
FF02:0:0:0:0:0:0:2
FF05:0:0:0:0:0:0:2
```

2.3.3.2.4. Solicited Node address

In order to make IPv6 address resolution (i.e. mapping the nodes' hardware addresses to IPv6 addresses) more efficient, a Solicited Node multicast address has been defined. This address is constructed from the 24 least-significant bits of a node's unicast or anycast address in the following manner:

```
FF02:0:0:0:0:1:FFXX:XXXX
```

For example, a node with a unicast address of :

```
1234:5678:9ABCD:EF12:FEDC:BA98:7654:3210
```

will join a multicast group identified by the Solicited Node address of:

```
FF02:0:0:0:0:1:FF54:3210
```

The rationale behind this address type is that within a local network, the least significant bits of the addresses of all the nodes are very likely to be different, so it is highly probable that a node will be the sole member of such a multicast group. *If the underlying data link*

protocol supports multicasting, like Ethernet does, the Neighbor Solicitation message (which is used for neighbor discovery) *is not broadcast. Instead, it is sent to the solicited-node address of the device whose IPv6 address we are trying to resolve.* [3.c] In other words, the address resolution procedure of IPv6 is more efficient in terms of network usage, as it is able to use the network's multicast capabilities to direct the message to the proper destination.

2.4. Address allocation

The table below lists the current allocation of the address space to various types of addresses. It should be noted that as of now only about 30% of the total address space is assigned to some use. [14]

Address type	Binary prefix	IPv6 notation
Unspecified	`000....000` (128 bits)	`::/128`
Loopback	`000....001` (128 bits)	`::1/128`
Multicast	`11111111` (8 bits)	`FF00::/8`
Link-Local unicast	`1111111010` (10 bits)	`FE80::/10`
Global unicast	*everything else*	

The Internet as a whole is a largely decentralized network, but address allocation is a task that requires a single central authority that would be responsible for proper management. Similarly as in the case of the IPv4 address space, the task of managing IPv6 addresses was given to the Internet Assigned Numbers Authority (IANA), that would delegate groups of addresses to Regional Internet Registries (RIRs). The RIRs will in turn delegate the addresses to other registries, ISPs and organizations. [19]

2.5. Special addresses

A number of special addresses have also been defined in IPv6. They are listed below.

2.5.1. The Unspecified address

The address of the form given below should never be assigned to any node and is used to indicate the absence of an address.

```
0:0:0:0:0:0:0:0
```

In a compact form, this address can be represented simply as:

```
::
```

2.5.2. The Loopback address

A loopback address is a unicast address that may by used by a node to send packets to itself. [14] This address has the following form:

```
0:0:0:0:0:0:0:1
```

which can be represented in compact form as:

```
::1
```

2.5.3. The Link-Local address

The Link-Local address is a unicast address indented for the use on a single link. Such an address can be automatically generated by a node using its hardware address (*Interface ID*). It has the following format:

10	54	64
1111111010	0	Interface ID

Figure 5: The Link-Local address format

Such an address can be used for communication between the nodes on a local network when no access to an external network is necessary.

2.5.4. The IPv4-Mapped IPv6 address

For the transition period when both IP protocols will have to coexist, an IPv6 address type was defined that carries an IPv4 address in the least-significant 32 bits.

80	16	32
0000...000	FFFF	IPv4 address

Figure 6: The IPv4-Mapped IPv6 address format

The use of these addresses is discussed in the section related to the Transition Mechanisms later in this document.

2.6. Required addresses

As one can observe from the previous sections, IPv6 defines quite a number of addresses that serve important functions in the network. A single IPv6 node must then be aware of many more address types in comparison with IPv4. The addressing specification [14] gives a detailed list of addresses that a node or a router must recognize as identifying itself:

- Node's addresses:
  - Link-Local address for each interface.
  - Manually configured unicast or anycast addresses.
  - The loopback address.
  - The All-Nodes multicast address.
  - The Solicited-Node multicast address.
  - Multicast addresses for all groups the node is a member of.
- Router's addresses:
  - All the addresses listed above - recognized by a node.
  - The Subnet-Router anycast address for each subnet.
  - The All-Routers multicast address.

3. Packet structure

The structure of the IPv6 packet is a modified version of the IPv4 packet. Two major changes were introduced: the main header was simplified, and extension headers may now follow the main header, carrying additional information. When analyzing the structure of the packet, one should take into consideration the design goals that resulted in such a solution, which were the reduction of the processing cost of packet handling at the routers, and to limit the bandwidth cost of the header. [4]

3.1. Header format

The structure of the IPv6 header is shown below. [4]

Figure 7: IPv6 header structure

The header is fixed in size and has a length of 40 bytes. As one can see, the major part of the header is comprised of the source and destination IPv6 addresses, which account for 80% of the total length (32 bytes). The remaining fields are described below:

- **Version** – A 4-bit Internet Protocol version number, set to 6. This is the only common field with the fourth version of the protocol, and is sufficient for both protocols to coexist in a single network.
- **Traffic class –** This 8-bit field is intended to be used for identifying and distinguishing between various classes or priorities of IPv6 packets. The IPv6 specification [4] does not provide any guidelines for the usage of this field and states that experiments are underway which will lead to an eventual specification.

- **Flow label –** A 20-bit field that *may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service.* [4] Similarly to the *traffic class* field, experiments are underway that will determine the usage of this field.
- **Payload length** – A 16-bit unsigned integer that specifies the length of the rest of the payload (excluding the main header, but including the extension headers) in octets. This leads to a maximum value of a packet to be 65,535 + 20 bytes.
- **Next header** – This field specifies the type of the header that immediately follows the main IPv6 header. This field uses the same values as the IPv4's *Protocol* field
- **Hop Limit –** The meaning of this field exactly the same as the IPv4's *Time To Live* (TTL) field, i.e. it specifies the maximum numbers of routers a packet may traverse. Each node that forwards the packet decrements the Hop Limit field by one. Once a value of zero is reached, the packet is discarded.

One may notice that the IPv6 header does not have a checksum field, which was present in case of IPv4. The reason for removing this field was to minimize the time required for the router to process a packet, as checksum calculation might be a time-consuming process, and to reduce the complexity of the routers themselves. The removal of the checksum field was possible as most of both the lower (Ethernet, etc.) and higher level (TCP, UDP, etc.) protocols have their own error detection capabilities.

3.2. Extension headers

In addition to the main IPv6 header, a packet may have additional headers, called *extension headers*, that carry information that is not contained in the main header. The headers are daisy-chained using the **Next header** field present in each of the headers.

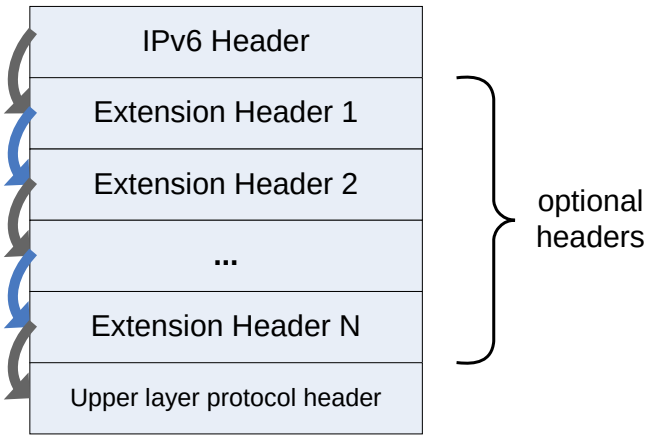

Figure 8: Daisy-chaining of extension headers

It is important to note that the extension headers are optional. They are only used if conveying additional information in the packet is necessary.

IPv4 also allows for optional information to be transmitted in the packet header by the use of the *Options* field, which has a variable length to encompass it. This approach was dropped, as Extension headers are a more flexible and robust solution.

Each header can have its own internal structure of fields. The headers are identified by the *Next header* field of the previous header. The last header in the chain is usually the upper layer protocol header, such as TCP or UDP.

3.2.1. Order of headers

To simplify routing and minimize the processing required at each intermediate node, the whole IPv6 packet needs to be processed only by the destination node. Some extension headers need to be read by the intermediate nodes along the path, though, as they might carry relevant information. That is why a certain order of extension headers is recommended [4].

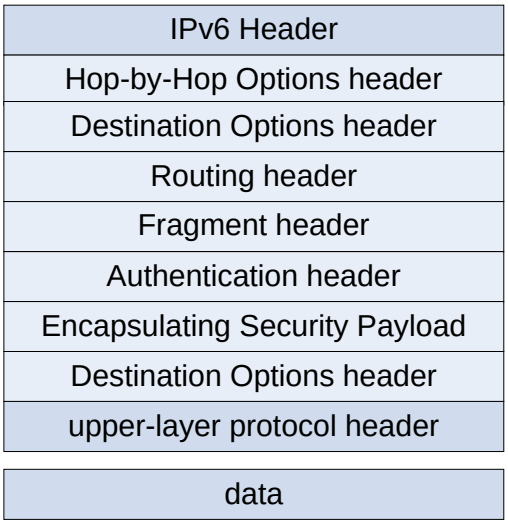

Figure 9: Recommended order of extenstion headers

This order is presented in the figure below. When a node processes a packet, it must analyze extension headers in the order that they appear, and not look for a particular header while ignoring the others. Only after an extension header has been processed may a node move to the next one.

Each extension header may appear in the packet at most once. The only exception to this rule is the *Destination Options* header, which may appear twice - before the *Routing* header to convey options that are to be processed by each destination specified in this header (see the section on the *Routing* header for more information), and for the second time just before the upper-layer protocol header to convey options intended for the ultimate destination of the packet.

A full IPv6 protocol implementation **must** support all of the extension headers listed above. This means that the security features offered by IPsec, through *Authentication* and *Encapsulating Security Payload* headers, are an integral part of IPv6.

3.2.2. Hop-by-Hop Options header

This header contains *an arbitrary set of options that are intended to be examined by all devices on the path from the source to destination device(s)* [3]. This is the only header that needs to be examined by all nodes along the path.

An example use of this header is to specify the *Jumbo Payload* hop-by-hop option, defined in RFC 2675 [21], which allows an IPv6 packet to carry a payload with a size within the range from 65,536 to 4,294,967,295 octets. Packets with such a large payload are called “jumbograms”.

3.2.3. Destination Options header

The Destination Options header carries options that are intended to be processed by the destination node. If the header is placed before the *Routing* header, then it is to be examined by all intermediate destinations specified in that header. If the header is placed before the

upper-layer protocol header, then it is indented to be processed by the ultimate destination node of the packet.

No practical usage examples are given in the specification for this header.

3.2.4. Routing header

The Routing extension header is used to specify a routing path. It has a function similar to the IPv4 Loose Source Route and Record Route options, i.e. it specifies an incomplete list of routers a packet must visit on its way to the destination.

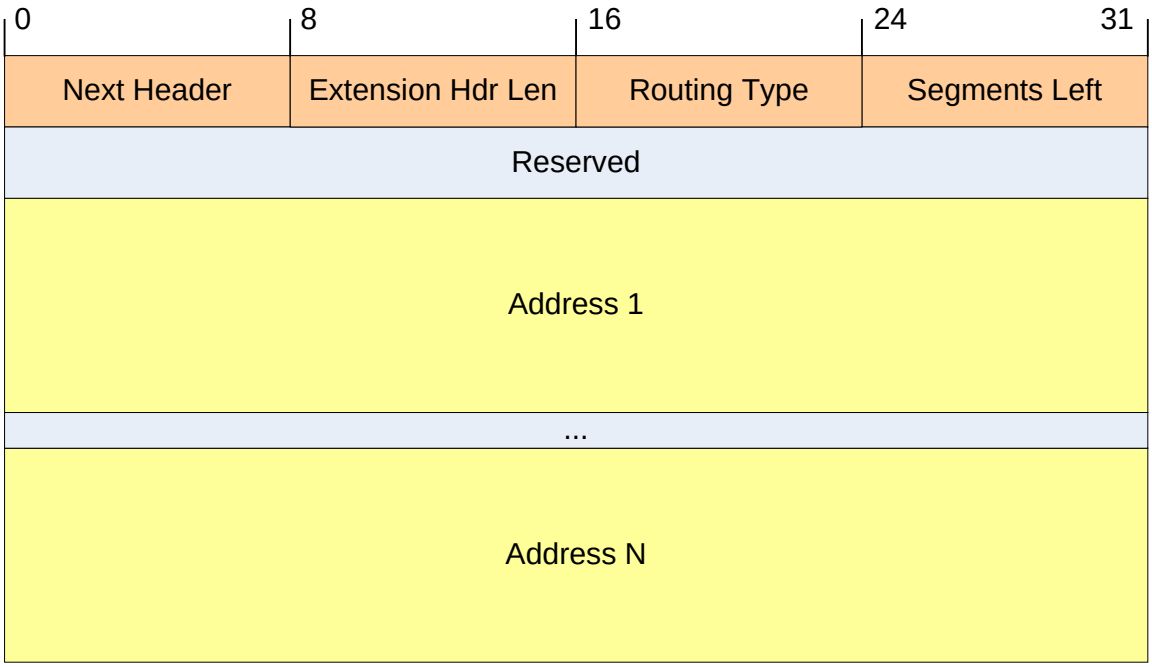

Figure 10: Routing extension header structure

The figure above presents a **Type 0** Routing header, the only one defined in [4]. It contains a loose list of addresses that a packet should follow.

The Routing header is examined only when the packet reaches the node specified in the *Destination* field of the main IPv6 header. The procedure used then is explained on the following example.

Suppose a source **S** sends a packet to destination **D** (Fig. 11)**,** and the packet should follow a route through nodes **A, B** and **C**. The sender **S** creates a packet with the main header's *Destination* field set to **A**, the Routing header's *Segments Left* field set to the number of path elements (3), and lists the addresses of the intermediate hops starting at the second one and giving the ultimate destination address **D** as the last address in the list.

At each hop, the packet's *Destination* address is swapped with the address specified in the Routing header at the position calculated from (*total number of addresses - Segments Left)* and the *Segments Left* field is decremented by one. If *Segments Left* is zero, there are no more intermediate nodes to be visited en route to the destination.

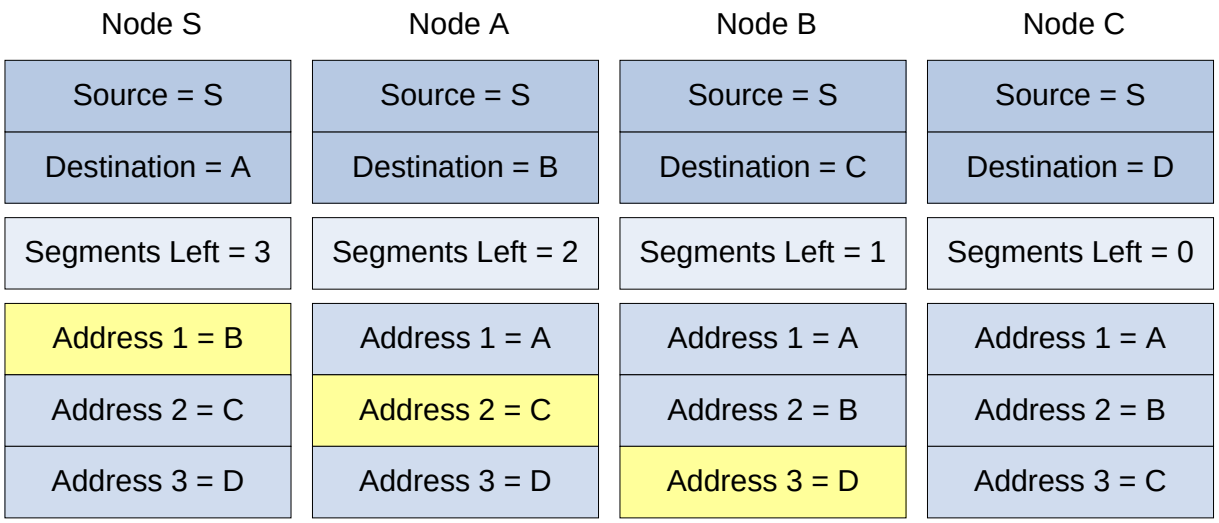

Figure 11: Loose Source Route procedure

3.2.5. Fragment header and packet fragmentation

The Fragment header is used to perform packet fragmentation in IPv6. Fragmentation of packets occurs if the length of the packet is too large to fit the *Maximum Transmission Unit* (MTU) of one of the networks along the packet's route.

3.2.5.1. Packet fragmentation

In IPv6 the original packet that is to be fragmented is divided into two parts. The **Unfragmentable part** consists of the main header and all extension headers up to and including the Routing header. This part is not fragmented and forms the basis of the fragmented packets. The **Fragmentable part** consists of all headers following the Routing header, including the upper-layer protocol header (e.g. TCP, UDP) and the data payload of the packet.

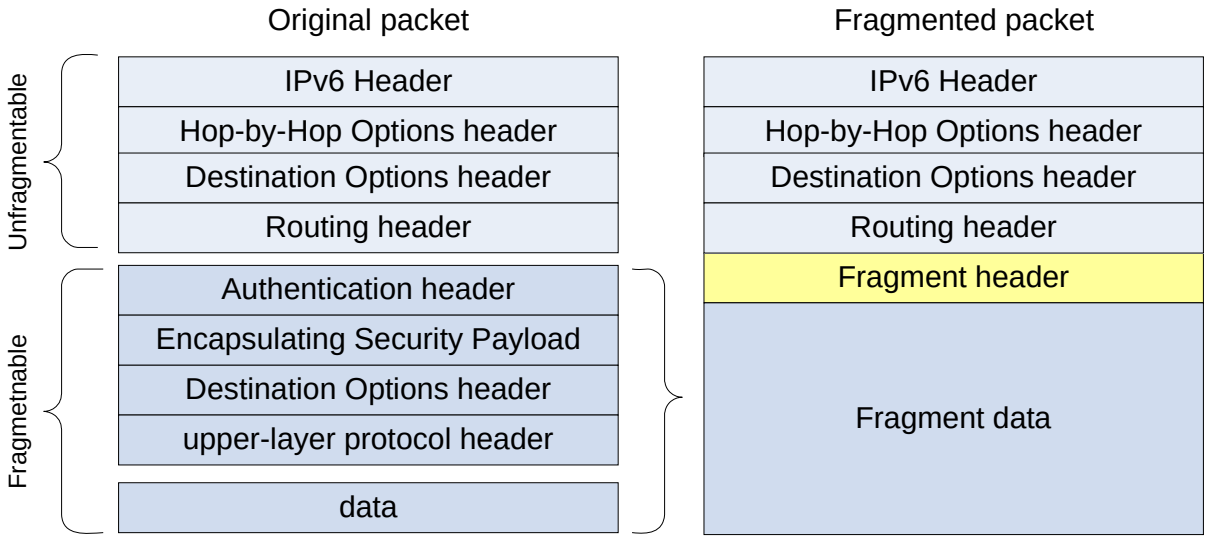

Figure 12: Packet fragmentation scheme

Each resulting fragment contains the **Unfragmentable part** of the original packet, the Fragment header, and a portion of the original packet's **Fragmentable part**. The headers of the fragmented packet are modified to form a proper packet, i.e. the IPv6 Header's *Payload*

Length field is changed to contain the length of this fragment only, and the last unfragmentable header's *Next Header* field is changed to the ID of the Fragment header. The Fragment header that is contained in each packet is also constructed to describe the particular fragment only.

The length of the fragments must be chosen in such a way as to fit the MTU of the path to the packet's destination.

In the scheme outlined above one can see why the order of extension headers is important. If the headers would be positioned randomly, then it would not be possible to distinguish the *fragmentable* and *unfragmentable* parts so easily.

3.2.5.2. Fragment header

The header itself has the following structure:

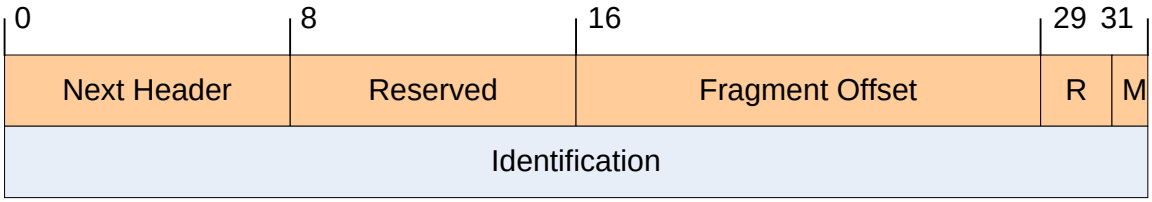

Figure 13: Fragment extenstion header

The 13-bit *Fragment Offset* field indicates the position relative to the start of the *Fragmentable part* in the original packet of the data following this header. The offset is given in 8-octet units. The **M**-bit is the equivalent of IPv4's *More Fragments/Last Fragment* field. If it is set to 1, then more fragments of this packet will be transmitted, otherwise it indicates that this is the last fragment of the original packet.

The 32-bit *Identification* field is set to a value that should be different for all fragmented packets with the same source and destination addresses that were recently sent by a node. Such a requirement is needed in order to prevent fragments of different packets to conflict with one another.

Two reserved fields are also present in the header and are marked in the above figure as *Reserved* and *R*.

3.2.5.3. Source-only fragmentation

Another distinct feature of IPv6 as compared to IPv4 is the notion of source-only fragmentation [4]. In IPv6 fragmentation may only occur at the source node and intermediate nodes are not allowed to perform packet fragmentation. If a packet received by a router is larger than the MTU of the interface it needs to be forwarded to, then an ICMPv6 [22] *Packet too big* message is to be sent back to the source and the packet is to be discarded. Such an ICMP message contains the allowed MTU of the interface and a portion of a packet that triggered the error message.

In order for a sending node to make sure that its packets will reach the destination without triggering a *Packet too big* message, the node can either send packets that are equal to or smaller than the minimal MTU specified for IPv6 [4], or invoke the Path MTU Discovery protocol.

The minimum MTU for all links that support IPv6 communication is defined to be 1280 octets. Each node is also required to be able to accept fragmented packets which after reassembly are as large as 1500 octets.

The Path MTU Discovery protocol, described in [23], is an optional protocol that can be implemented (and its implementation is, in fact, recommended by RFC 1981) by IPv6 nodes

to *discover and take advantage of paths with PMTU greater than the IPv6 minimum link MTU. A minimal IPv6 implementation (e.g., in a boot ROM) may choose to omit implementation of Path MTU Discovery* [23]. The implementation of the protocol is made optional as some devices (e.g. embedded devices) might never send packets larger than the minimal MTU, and therefore have no need for this protocol.

The two most important implications arising from such an approach are that the routers can process packets faster, and that the route between the source and destination must remain reasonably static or otherwise lots of ICMP *Packet too big* messages might be generated.

3.2.6. Authentication and ESP headers

The Authentication and Encapsulating Security Payload headers are cryptographic protocols for securing packet flows that are a part of the IPsec (IP security) standard [24]. They are already used in conjunction with IPv4 to provide network layer security, but as of IPv6 they are an integral part of the IP protocol.

These protocols can operate in various modes offering authentication, data confidentiality and message integrity. A key exchange protocol (IKE, Internet Key Exchange) is also a part of the IPsec standard.

4. Transition Mechanisms

The problem of transition from IPv4 to IPv6 is a very important factor in the protocol's development. The transition process involves publishing recommended ways for hosts to switch to the new protocol, creation of compatible versions of operating systems and application software, requires availability of supporting network hardware, and many other issues that need to be resolved.

A special IETF working group, called *IPng transition* or *ngtrans,* was formed to facilitate the transition process [25]. The group has set out certain milestones that would advance the protocol's deployment. Numerous RFC documents have been published that describe ways in which hosts can achieve IPv6 connectivity. Since 2003, the *ngtrans* group has been replaced by the *v6ops* (IPv6 Operations) working group that is more focused *on outlining transition scenarios and identifying the specific tools (many from the ngtrans effort) that will be used in a transition.*

As of 2006, all of the popular operating systems (Windows, Linux, MacOS, BSD) support IPv6 [26]. Support in client and server software for this protocol is also very strong [27] and it is possible to deploy virtually any Internet service (Web, email, ftp, secure shell, instant messaging, VoIP, etc.) already available through IPv4 on the IPv6 network.

One might say that the only thing lacking now is the will to actually make the transition. I have described some of the issues that hinder this process earlier in this document, the prime problem being the lack of backward compatibility which would allow the transition to be very smooth and unnoticeable to most of the users. Unfortunately, this seems to be not possible due to technical reasons.

The summarize, in order to communicate over IPv6 one requires:

1. An IPv6 compatible operating system, i.e. containing an IPv6 protocol stack.
2. IPv6-compatible application software, as some programs that depend on distinct features of IPv4 might not work over IPv6. One such group of software are programs that use protocols which embed IP addresses in their payload, e.g. FTP clients and servers.
3. A connection to the IPv6 network from an Internet Service Provider.

The third point is currently the major problem, as IPv6 connectivity is not popular among ISPs and it could be difficult to find one in a desired area that offers such a service. Furthermore, it is not plausible to connect only to the IPv6 network without being able to communicate with IPv4 hosts, as currently the global Internet relies mostly on IPv4.

In the following sections I have described some general approaches to achieving connectivity to the IPv6 network.

4.1. Dual IP Stack

The most straightforward way to connect to the IPv6 network without losing IPv4 connectivity is the Dual IP Stack approach (sometimes referred to as *Dual IP Layer*[1]). In this scheme a host is connected directly to both the IPv6 and IPv4 networks, has both type of addresses assigned to itself, and is capable of communicating natively with both types of networks. [29]

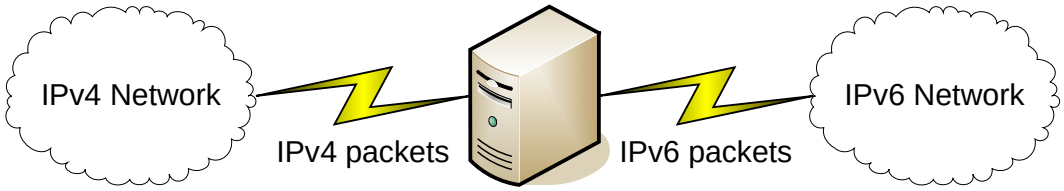

Figure 14: A Dual Stack host

According to [28], most of the current IPv6 implementations are dual stack and there is in fact no known implementation which is a IPv6-only one.

The operation of the Dual Stack implementation is illustrated below. One can distinguish four cases of operation: [30]

a) An IPv4-only application tries to communicate with an IPv6 destination.
b) An IPv6-only application tries to communicate with an IPv4 destination.
c) An IPv4 (IPv6) application tries to communicate with an IPv4 (IPv6) destination.
d) A IPv4 and IPv6 compatible application tries to communicate with either IPv4 or IPv6 destination.

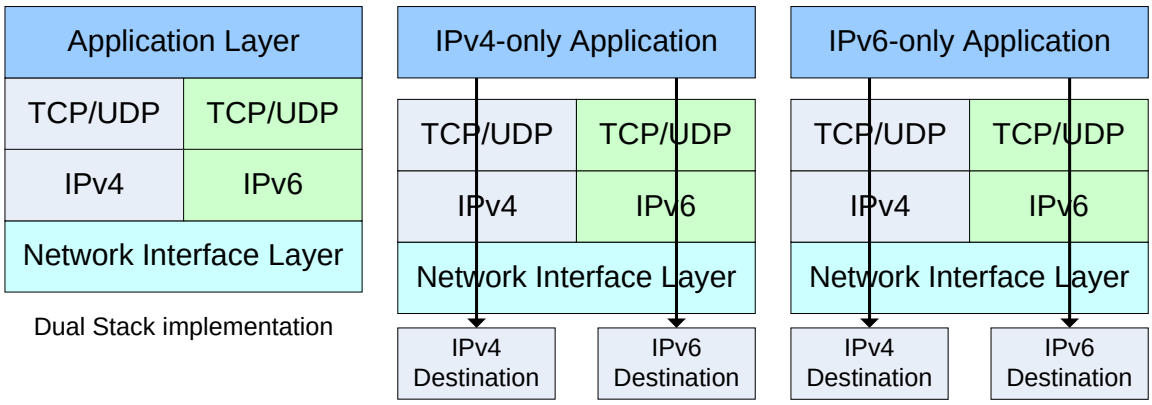

Figure 15: Dual Stack implementation and operation

Cases *c)* and *d)* are uninteresting as an application that was designed for a certain protocol can obviously communicate using this protocol, and protocol-independent applications have similarly no difficulties in communicating using any protocol (the only issue in this case is proper DNS name resolution, which will be discussed later).

[1] Microsoft's TechNet article [32] makes a discrimination between the term "Dual IP Stack" and "Dual IP Layer". The former one is suggested to mean an architecture in which both the Network and Transport Layer protocols are implemented separately (i.e. not only separate IPv4/IPv6 code, but also separate code for TCP, UDP, etc. protocols), while the latter is an architecture where only the IP Layer protocol is a separate piece of code. This division is quite vague, as the IPv6 protocol requires minor changes in TCP and UDP protocols (e.g. changes in checksum calculation), so actually IPv4's TCP is incompatible with IPv6's TCP.

An IPv6-only application can communicate with an IPv4 destination (Fig. 16) by using IPv4-mapped IPv6 addresses, discussed earlier in this document, which embed the host's IPv4 address within an IPv6 address. In this way, the IPv6-only application establishes a connection to the IPv4-mapped IPv6 address and the host's Dual IP stack can then use the IPv4 module to establish the actual connection in a way that is transparent to the application.

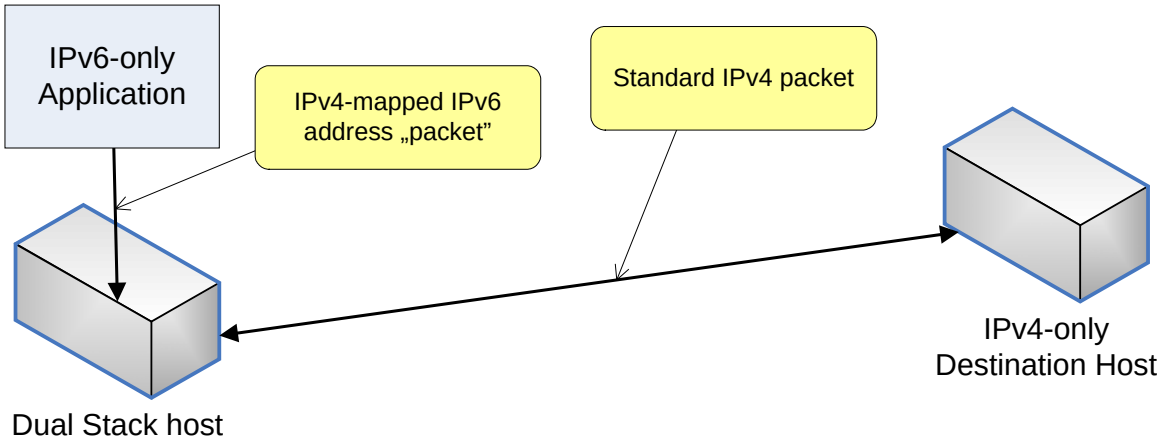

Figure 16: IPv6-only application to IPv4-only host communication

A more complicated scenario is the case when an IPv4-only application tries to establish a connection with an IPv6 destination. The recommended approach in this case, according to [29], is to port the application to IPv6. If this is not possible (e.g. due to unavailability of the application's source code) certain solutions can be applied, but they are not guaranteed to work with all types of applications. One of the proposed solutions is called *Bump-in-the-Stack (BIS)*, which involves a special module in the IPv4 stack that would intercept packets and translate IPv4 addresses to IPv6 and vice versa [31], similarly to *Application Layer Gateways (ALGs)* in *Network Address Translation (NAT)* systems.

The Dual Stack approach has the following advantages:

- Dual-stack hosts can reach both networks.
- Application compatibility is generally maintained.
- Connectivity with IPv4 hosts is preserved.

4.2. Configured tunneling

Another approach to IPv6 connectivity is configured tunneling. This method can be used when two IPv6 networks are separated by an IPv4 network. Such a scenario is expected to be the dominant case in the transition period, i.e. there will be a large number of IPv6 networks interconnected by the global IPv4 Internet. [29]

Figure 17: IPv6 networks connected via IPv4 network

In configured tunneling, IPv6 communication between two networks is established by manually set up tunnels. These tunnels need to be configured by the network administrators at both sites, and IPv6 packets are then forwarded over the IPv4 network using encapsulation.

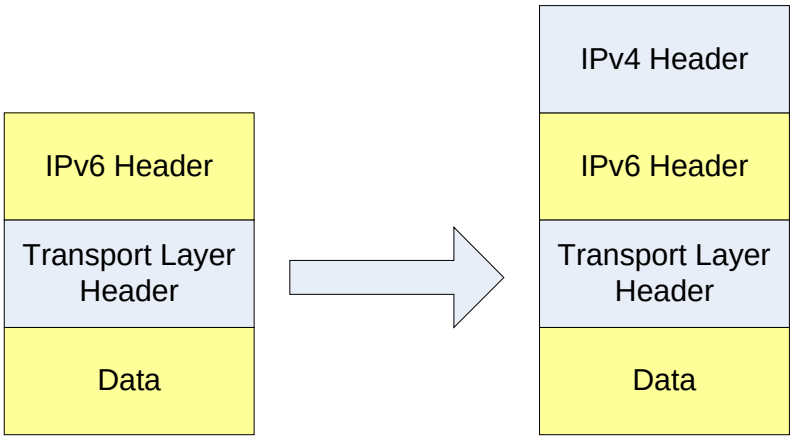

Figure 18: IPv6 packet encapsulation in IPv4

The communication scheme is quite simple. The sender of a packet destined for a host in the other network creates a properly addresses normal IPv6 packet. This packet is sent to the default gateway (the border router), which operates the tunnel endpoint. The border router encapsulates the IPv6 packet inside an IPv4 packet destined to the other tunnel endpoint (identified by an IPv4 address) by prepending an IPv4 header. Such a packet traverses the IPv4 network as any other packet. At the border router of the destination network, the IPv4 header is stripped and the packet is forwarded as a normal IPv6 packet.

As manual configuration is involved in setting up such tunnels, the addresses of tunnel endpoints are determined from stored configuration files. Some security measures can also be involved in the tunnel setup. For example, the routers can reject packets which originate from an invalid source address. Furthermore, tunnel encryption can also be established.

The configured tunneling method offers a great deal of flexibility due to the fact that it involves a great deal of human interaction to establish such tunnels. This is also its primary weakness, as it is infeasible to set up hundreds or thousands of such tunnels which may be required to establish connectivity with all IPv6 networks.

4.3. Automatic tunneling

The transition methods described in the previous section have one important drawback. They do not account for the situation where a site (or a host) does not have any particular network to connect to, but wishes to establish communication with the global IPv6 network. As I mentioned earlier, in the early stages of IPv6 transition it could be difficult to obtain a direct connection to the IPv6 Internet from an ISP. This problem is partly solved by automatic tunneling.

The automatic tunneling scheme enables any site that already has a globally unique IPv4 address to construct its own set of IPv6 addresses. *IPv6 sites or hosts connected using this method do not require IPv4-compatible IPv6 addresses or configured tunnels. In this way IPv6 gains considerable independence of the underlying wide area network and can step over many hops of IPv4 subnets. The abbreviated name of this mechanism is 6to4.* [33]

4.3.1. Prefix assignment

The Internet Assigned Numbers Authority (IANA) has permanently assigned a 16-bit prefix to this scheme:

```
2002::/16
```

which can be used to construct a prefix for a site using a single IPv4 address:

16	32	16	64
2002	IPv4 address	SLA ID	Interface ID

Figure 19: 6to4 address format

where *SLA ID* is the Site Level Aggregation identifier as described in the original addressing scheme earlier in this document (RFC 3056 [33] which introduces automatic tunneling was written before this addressing scheme was abandoned). The resulting prefix will have the following form:

```
2002:IPv4ADDR::/48
```

For example, a site which has a globally unique IPv4 address of *123.234.012.156* would be in possession of the following prefix:

```
2002:7BEA:C9C::/48
```

Such a prefix can then be used to subnet the site's network and assign individual addresses to hosts.

4.3.2. Connectivity scenarios

Automatic tunneling involves the dynamic construction of tunnels between IPv6-enabled networks. This requires that the addresses of the tunnel endpoints be known, or more specifically, be determined in some automated manner.

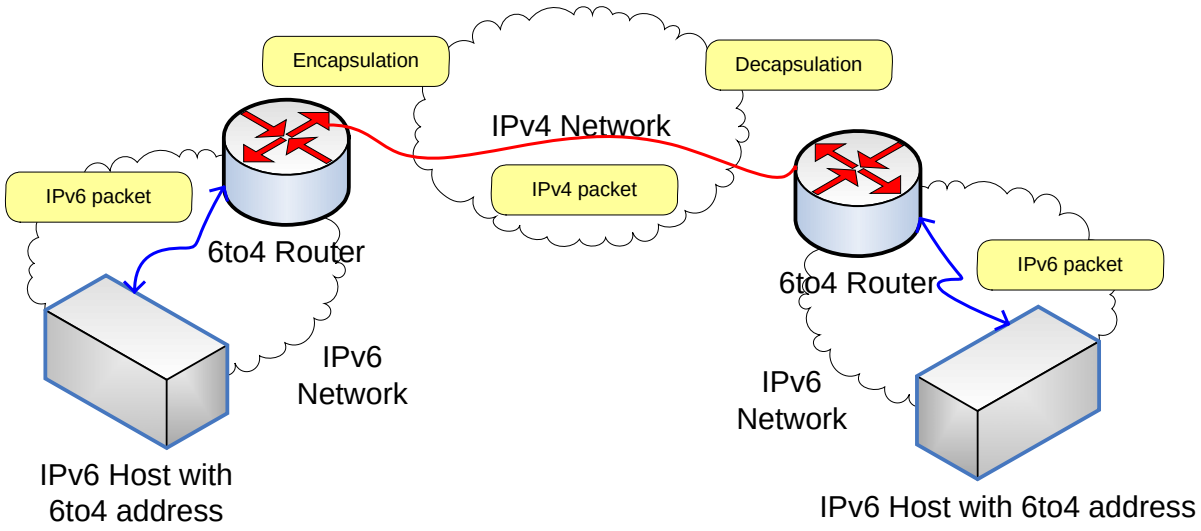

Figure 20: 6to4 communication scheme

We can distinguish two interesting cases with regard to tunnel endpoint address determination, which are described later on. Apart from that, communication in the 6to4 scheme takes places in the following way:

1. A IPv6 host generates a packet to another IPv6 destination.
2. The packet is forwarded to the border router of the network, which establishes a tunnel over the IPv4 network to the destination's border router. The IPv6 packet is encapsulated in IPv4.
3. The destination's border router decapsulates the packet and forwards it as a normal IPv6 packet.

4.3.2.1. 6to4-only to 6to4 and native-IPv6 scenario

When a 6to4-only site wants to send a packet to a site that has both 6to4 and native IPv6 connectivity, it must use the destination site's 6to4 address, as this address contains the IPv4 address that is required to set up a tunnel. A 6to4-only site cannot directly communicate with a site that has only a native IPv6 address.

Similarly, a site that has both 6to4 and native-IPv6 addresses must use its 6to4 address when communicating with a 6to4-only site.

4.3.2.2. 6to4-only to native-IPv6-only scenario

Communication between a 6to4-only site and a IP6-only site is possible through the use of 6to4 *relay routers*.

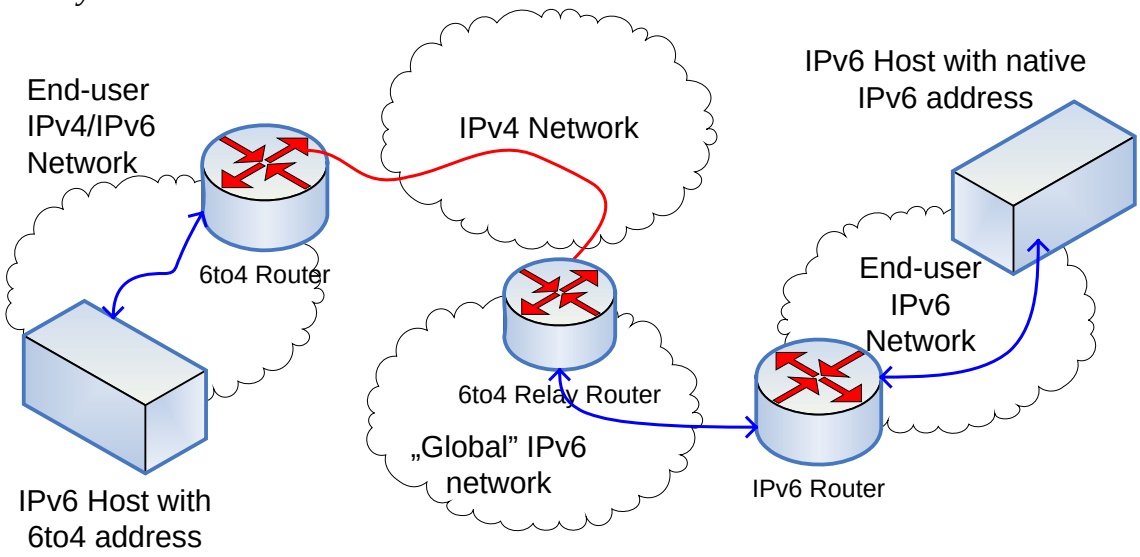

Figure 21: 6to4 communication with relay routers

A 6to4 relay router is a router that has both 6to4 and native IPv6 addresses. There is, in fact, nothing special about such router except for the addresses it needs and its support for the 6to4 pseudo-interface. The model of communication is similar to the one described in the previous section.

The question that arises from the above description is: how to obtain the addresses of the 6to4 relay routers? The solution to this problem has been provided by RFC 3068 [34], which specifies a *6to4 IPv6 relay anycast address*:

```
2002:c058:6301::
```

6to4 routers can use the above address as the default route for native IPv6 traffic. This anycast address has the form of a normal 6to4 address, i.e. it holds an embedded IPv4 address. In this case, the address is *192.88.99.1*. The *anycast* function is realized in the IPv4 network through various routing protocol setups. [34]

4.4. Other considerations

4.4.1. DNS

The Domain Name System's (DNS) role has become more significant with the advent of IPv6 as longer addresses make it very hard to input them "by hand". Furthermore, in the transition period the applications will rely on DNS to guide them to a host using the appropriate protocol.

4.4.1.1. New DNS records

The primary services of the DNS to the IP network remain the same. A few changes to the system have been introduce to accommodate data associated with IPv6 addressing:

- An AAAA record – which is the equivalent of the A record. [35], but is now considered obsolete.
- An A6 record – which is a more robust equivalent of the A record, supporting indirect addressing and renumbering (in IPv6 renumbering a network should be much easier than in IPv4). [36]
- A new domain - IP6.ARPA – to support reverse lookups of addresses. [36]

4.4.1.2. Proper DNS resolution for applications

Some applications will depend on a proper type of address being returned from a DNS query during the transition period. This involves applications that require either IPv4 or IPv6 to operate properly, or applications that try to communicate over certain schemes that establish connectivity, like 6to4. These applications will not be able to cope with addresses of a different type that they expect. This problem has been solved by letting the application specify the type of address that it would prefer to receive.

5. Conclusion

The sixth version of the Internet Protocol provides many advantages over the previous one. It not only resolves the problem of address space exhaustion, but also introduces improvements that will make managing networks easier and more effective. By moving complexity away from the routers to the network endpoints it tries to make the routing infrastructure simpler and more efficient. One of the major problems of current day routers – the overwhelming number of routing prefixes – has been solved by introducing hierarchical addressing. Enhanced autoconfiguration features make connecting new devices to the network an easy task as many parameters can be determined automatically. IPv6 has also provisions for security and Quality of Service. To make the protocol easily adaptable to future uses, the packet structure is able to carry additional information through the use of extension headers.

Despite the quite impressive list of advantages, IPv6 is still not widely used today. The primary reason is the fact that the protocol is not backwards compatible. This results in many complex transition scenarios that need to be dealt with for the protocol to be a success. The ongoing work on the transition mechanisms is quite successful in areas ranging from backbone network operators, through regional networks, to local ISPs and larger organizations. That is, all the necessary components (hardware, software) and guidelines for transition are available. The thing that is lacking now is the will to actually perform the transition. One of the reasons that organizations (and major services, like Google, CNN, etc.) are not keen on switching to IPv6 is that their customers are not there yet. These customers are the majority of people using the Internet – connecting via DSL, Cable or an "old fashioned" dial-up modem. These "ordinary" users are probably the most difficult problem in the entire transition, as switching them to IPv6 requires updating millions of home computers. And this not only means updating the software, which, for the most part, is already IPv6-compatible and if not, can usually be updated automatically, but reconfiguring the systems to run IPv6. On a Windows machine, this means installing and configuring a new TCP/IP protocol driver – a task that might be too complex for the average user. This is an issue that needs to be resolved before IPv6 can be widely deployed.

To conclude, one can expect that adoption of IPv6 will continue and one day in the near future (I would say it is rather a matter of a 5 to 10-year period) it will reach the stage of being the dominant Network Layer protocol of the world.

6. References

The list of references is presented below. Note that I did not provide URLs for RFC documents, as these documents are widely available on the Internet from many sources. One of them is the Internet Engineering Task Force's Request for Comments repository at http://www.ietf.org/rfc.html.

1. "RFC 791: Internet Protocol. DARPA Internet Program Protocol Specification", September 1981.
2. "IPv6 - Why does the Internet need a new protocol?", 1997, http://www.luv.asn.au/overheads/ipv6/ipv6_why.html [accessed: May 24, 2006]
3. „The TCP/IP Guide: Internet Protocol Version 6 (IPv6) / IP Next Generation (IPng)", http://www.tcpipguide.com/free/t_InternetProtocolVersion6IPv6IPNextGenerationIPng.htm [accessed: May 24, 2006]
    a. http://www.tcpipguide.com/free/t_IPv6MotivationandOverview-3.htm
    b. http://www.tcpipguide.com/free/t_IPv6GlobalUnicastAddressFormat-3.htm
    c. http://www.tcpipguide.com/free/t_TCPIPAddressResolutionForIPVersion6.htm
4. "RFC 2460: Internet Protocol, Version 6 (IPv6) Specification.", S. Deering, R. Hinden, December 1998.
5. "RFC 1752: The Recommendation for the IP Next Generation Protocol", S. Bradner, A. Mankin, January 1995.
6. "RFC 1550: IP: Next Generation (IPng) White Paper Solicitation", S. Bradner, A. Mankin, December 1993.
7. "Internet-Draft. IP Next Generation Overview", R.M. Hinden, October 1994, http://www.sobco.com/ipng/archive/ipv6/hinden-ipng-overview-00.txt [accessed: May 24, 2006]
8. "RFC 1883: Internet Protocol, Version 6 (IPv6) Specification", S. Deering, R. Hinden, December 1995.
9. „The IPv6 mess", D.J. Bernstein, http://cr.yp.to/djbdns/ipv6mess.html [accessed: Apr 24, 2006]
10. "RFC 1631: The IP Network Address Translator (NAT)", K. Egevang, P. Francis, May 1994.
11. "RFC 1519: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", V. Fuller, T. Li, J. Yu, K. Varadhan, September 1993.
12. "Could a U.S. Shift to IPv6 Cost $75B?", S. M. Kerner, *Internetnews.com*, http://www.internetnews.com/infra/article.php/3570211 [accessed: Apr 24, 2006]
13. "Age of the Universe", E. L. Wright, http://www.astro.ucla.edu/~wright/age.html [accessed: May 25, 2006]
14. "RFC 4291: IP Version 6 Addressing Architecture". R. Hinden, S. Deering. February 2006.
15. "Dot-decimal notation", *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Dot-decimal_notation [accessed: May 25, 2006]
16. "RFC 2374: An IPv6 Aggregatable Global Unicast Address Format", R.Hinden, M. O'Dell, S. Deering. July 1998.
17. "RFC 3587: IPv6 Global Unicast Address Format". R. Hinden, S. Deering, E. Nordmark. August 2003.
18. "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", http://standards.ieee.org/regauth/oui/tutorials/EUI64.html [accessed: May 25, 2006]

19. “RFC 1881: IPv6 Address Allocation Management”. IAB and IESG. December 1995.
20. “RFC 1924: A Compact Representation of IPv6 Addresses”. R. Elz. April 1996.
21. “RFC 2675: IPv6 Jumbograms”. D. Borman, S. Deering, R. Hinden. August 1999.
22. “RFC 4443: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification”. A. Conta, S. Deering, M. Gupta. March 2006.
23. “RFC 1981: Path MTU Discovery for IP version 6”. J. McCann, S. Deering, J. Mogul. August 1996.
24. “IPsec”, *Wikipedia, the free encyclopedia,* http://en.wikipedia.org/wiki/IPSec [accessed: May 27, 2006]
25. “ngtrans Home Page”. http://www.6bone.net/ngtrans/ [accessed: May 27, 2006]
26. “IPv6 in Operating Systems”, http://www.join.uni-muenster.de/Implementationen/Betriebsysteme.php?lang=en [accessed: May 27, 2006]
27. “IPv6 in different software“, http://www.join.uni-muenster.de/Implementationen/Software.php?lang=en [accessed: May 27, 2006]
28. “IPv6”, *Wikipedia, the free encyclopedia,* http://en.wikipedia.org/wiki/IPv6 [accessed: May 28, 2006]
29. “RFC 4213: Basic Transition Mechanisms for IPv6 Hosts and Routers”. E. Nordmark, R. Gilligan. October 2005.
30. “RFC 4038: Application Aspects of IPv6 Transition”. M-K. Shin, Ed., Y-G. Hong, J. Hagino, P. Savola, E. M. Castro. March 2005.
31. “RFC 2767: Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)”. K. Tsuchiya, H. Higuchi, Y. Atarashi. February 2000.
32. “How IPv6 Works”, *Microsoft TechNet,* http://technet2.microsoft.com/WindowsServer/en/Library/9bcf5d01-a1df-4053-939b-904e207535531033.mspx?mfr=true [accessed: May 28, 2006]
33. “RFC 3056: Connection of IPv6 Domains via IPv4 Clouds”. B. Carpenter, K. Moore. February 2001.
34. “RFC 3068: An Anycast Prefix for 6to4 Relay Routers”. C. Huitema. July 2001.
35. “RFC 1886: DNS Extensions to support IP version 6”. S. Thomson, C. Huitema. December 1995.
36. “RFC 2874: DNS Extensions to Support IPv6 Address Aggregation and Renumbering”. M. Crawford, C. Huitema. July 2000.