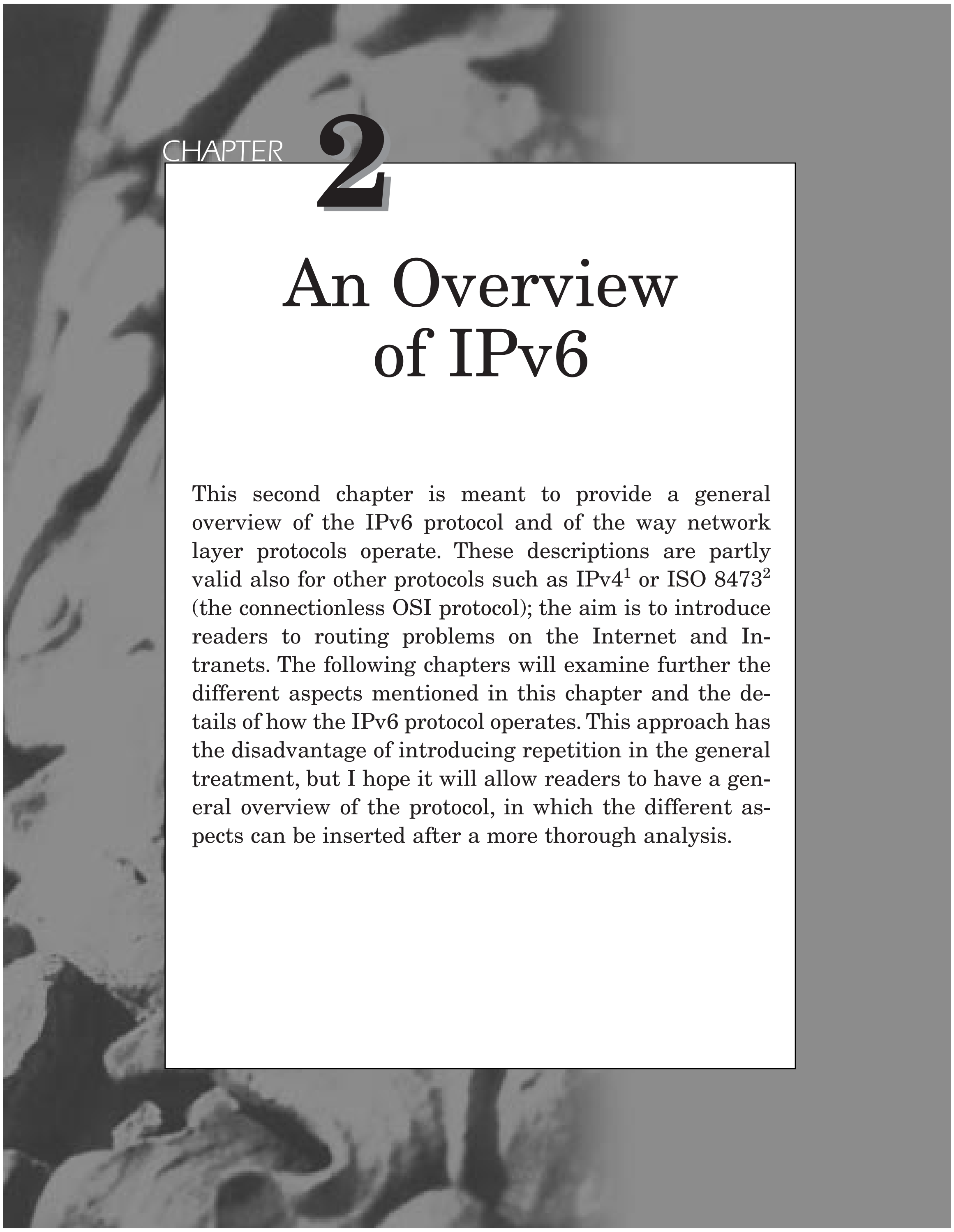


CHAPTER 2

An Overview of IPv6

This second chapter is meant to provide a general overview of the IPv6 protocol and of the way network layer protocols operate. These descriptions are partly valid also for other protocols such as IPv4[1] or ISO 8473[2] (the connectionless OSI protocol); the aim is to introduce readers to routing problems on the Internet and Intranets. The following chapters will examine further the different aspects mentioned in this chapter and the details of how the IPv6 protocol operates. This approach has the disadvantage of introducing repetition in the general treatment, but I hope it will allow readers to have a general overview of the protocol, in which the different aspects can be inserted after a more thorough analysis.

2.1 Terminology

Before discussing the treatment of IPv6, let me introduce terms used in standards[3]:

- *node*: A device that implements IPv6.
- *router*: A node that forwards IPv6 packets not explicitly addressed to itself.
- *host*: Any node that is not a router.
- *upper layer*: A protocol layer immediately above IPv6—for example, transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, or lower layer protocols being tunneled over IPv6 such as IPX and AppleTalk.
- *link*: A communication facility or medium over which nodes can communicate at the Data Link layer—that is, at layer 2 of the ISO/OSI reference model. Examples of links are Ethernet, PPP, X.25, Frame Relay, and ATM, or tunnels over other protocols such as IPv4 or IPv6 itself.
- *neighbors*: Nodes attached to the same link.
- *interface*: A node's attachment to a link.
- *address*: An IPv6 layer identifier for an interface or a set of interfaces.
- *packet*: An IPv6 layer PDU (Protocol Data Unit)—that is, the IPv6 header plus the payload.
- *datagram*: A synonym for *packet*.
- *link MTU*: The Maximum Transmission Unit—that is, the maximum packet size in octets (bytes) that can be conveyed unfragmented over a link.
- *path MTU*: The minimum link MTU of all the links in a path between a source node and a destination node.

2.2 Architecture of a Network

The terminology introduced in the preceding sections allows us to understand that, in general, an IPv6 network will be formed by a certain number of *routers* interconnected with a partially meshed topology, as shown in Figure 2-1.

56982_CH02I 12/12/97 3:29 PM Page 25

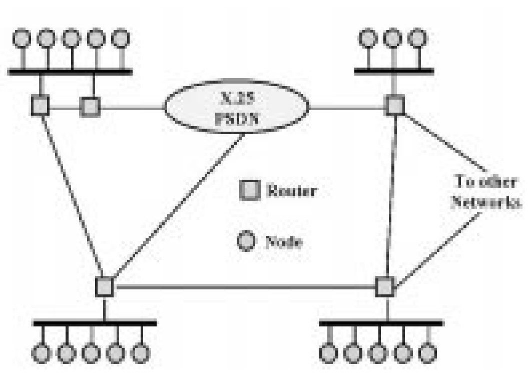


Figure 2-1
An example of a network

The choice of a partially meshed topology is justified by reasons of reliability. In fact, the mesh has alternative paths that can be used in case of fault. *Hosts* are generally interconnected to routers through *LANs* (local area networks)[4].

2.3 Addresses and Names

To reach all nodes in a network, the first problem to be solved is the unique identification of each node. IPv6 assigns a 128-bit numerical *address* to each network interface[5]. Nevertheless, in most cases, users find referring to a node using a *name* more convenient than using *a numerical address*. The name and the address of a system have the same purpose: the unique identification of an interface within the network. Nevertheless, the address is thought to interact with routing mechanisms and is therefore numerical, whereas the name is thought to be more easily remembered by the users and is therefore alphanumerical and mnemonic. Maintaining a biunivocal relation between names and addresses is clearly necessary, and doing so is more complex than one might think. In fact, in a small network, each computer maintaining a file with this relationship is foreseeable, but with the growth of network sizes, adopting a distributed database, called DNS (*Domain Name Service*), is essential[6].

If we want to use IP to build a worldwide computer network like the Internet, the addresses must be unique at the worldwide level. This requirement was already met by IPv4 addresses, but IPv6 extends the addresses to cope with the growth of the Internet and Intranets. This uniqueness is typically obtained through organizations that assign sets of addresses to end users.

These sets are called *networks* in IPv4 and can be subdivided into smaller sets, called *subnetworks*, through a parameter called a *netmask*. IPv4 requires that each link be associated to a subnetwork* so that checking whether two nodes are connected to the same link is easy; they are connected if their IP addresses belong to the same subnetwork.

In IPv6, the address organization is similar, but with two important differences:

- Addresses are longer (128 bits in IPv6 compared to 32 bits in IPv4).
- The concept of netmask is replaced by the concept of prefix. The prefix indicates how many bits are used to identify the subnetwork.

For example, in an IPv6 address with a prefix equal to 80, 80 bits will be used to identify the subnetwork and 48 bits to identify nodes within the subnetwork.

2.4 Routers and Internetworking

When a user wants to use an application on a given computer, that user can request it on the network by specifying the name of the computer; the network consults the Domain Name Service and extracts the IPv6 address of the remote computer. The address of the destination computer becomes the key element to determine the most suitable routing to reach the remote node. A first check made by the sender is whether the destination is connected to the same physical network of the sender; in this case, the transmission can occur directly. In the opposite case, an operation of *internetworking* is essential; the sender forwards the packet, and the router attends to its delivery.

The router's main task is precisely to route messages on the network. The chosen routing technique depends on the adopted network architecture. Connectionless protocols, such as IPv4, IPv6, IPX, DECnet, OSI-CLNP, and so on, use a technique known as *routing by network address*. A node is addressed by writing in the layer 3 packet (ISO/OSI reference model) its address, which must be unique on the network. Each router

*As a matter of fact, many IPv4 implementations release this original constraint, that it is preferable to observe to obtain better performance; this constraint has been reintroduced in IPv6.

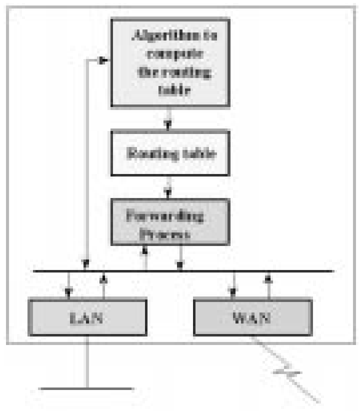


Figure 2-2
Routers internal architecture

uses this address as an index in its routing table and determines the path on which the packet must be retransmitted.

At this point, the important role of the *routing table* present on routers should be explained (see Figure 2-2).

When a packet reaches a router through a local or a geographical network interface, the router passes the packet to its *forwarding process,* which extracts the source address, uses this address to examine the routing tables, and decides on which interface to retransmit the packet.

2.5 The Routing Table

The routing table of an IPv6 router contains one *entry* for each subnetwork reachable from the router itself. A general scheme for a routing table organization[7] is shown in Figure 2-3. Routing tables can be written manually or computed automatically by appropriate protocols such as RIP[8] or OSPF[9].

In the example shown in Figure 2-3, we decided to use the name of the *subnetwork* itself, not its extended address. In the case of IPv6, for example, an address of the type **`FEDC:BB87:0:0:0:0:0:0/80`**, which is the address of a subnetwork with an 80-bits prefix (the syntax of IPv6 addresses will be explained in Chapter 4), can be associated to the name Delta.

Likewise, for the *Next Hop* field, for example, the Router-4 could have address **`FEDC:BB87:0:0:0:0800:2B3C:4D73`**.

The *Type* field indicates the type of reachability associated to the subnetwork. *Direct* indicates that the router has an interface directly connected to the subnetwork; *Static* indicates that a routing rule to reach the

Figure 2-3
Example of a routing table

Subnetwork	Next Hop	Type	Cost	Age	Status
Alpha	-	Direct	1	-	UP
Tau	-	Direct	1	-	DOWN
Beta	-	Direct	1	-	UP
Delta	Router-27	RIP	10	27	UP
Omega	Router-5	OSPF	5	13	UP
Gamma	Router-4	Static	2	-	UP

subnetwork has been written manually; *RIP* and *OSPF* indicate that the subnetwork reachability has been learned by the router through an appropriate protocol.

The *Age* field specifies the left validity in seconds, and it is significant only for entries associated to subnetworks whose reachability information has been learned through protocols for the automatic computation of the routing table. In fact, dynamic entries must be periodically updated.

The *Status* field indicates the entry's state. In our example, the router interface associated to the subnetwork Tau is down; therefore, the associated reliability information is not usable.

The router forwarding process uses the routing table for each packet by searching in the subnetwork column for which subnetwork the destination address belongs and then by routing the packet to the associated Next Hop. Note that Direct entries don't have a Next Hop because the router has an interface directly connected to those subnetworks and can therefore directly reach all the subnetwork nodes by link layer (also called layer 2 or Data Link layer) transmission (IPv6 terminology).

2.6 Layer 2 and Layer 3 Addresses

Until now, we have referred to 128-bit IPv6 addresses, corresponding to ISO/OSI reference model layer 3 or network layer addresses. Nevertheless, when a packet must be routed on a subnetwork, the transmission must occur at layer 2, which is at the link layer. Therefore, we must know and use layer 2 addresses. In the case of LANs, these addresses are the 48-bit MAC addresses; in the case of ATM, the 20-octet ATM addresses; and in the case of the point-to-point channels, they do not exist.

The need for two types of addresses can be summarized as follows:

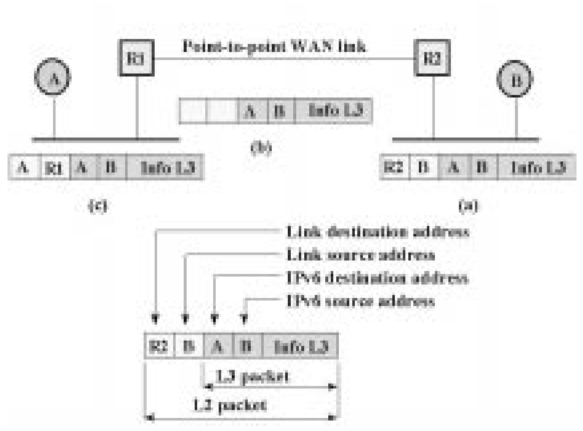
- The link layer address is used to identify the final destination of a packet within a physical network (IP subnetwork).
- The layer 3 address is used to identify the final destination of a packet within the whole network.

Different methodologies are available to maintain the mapping between link layer addresses and layer 3 addresses within a subnetwork. The best known is based on the ARP (Address Resolution Protocol)[10], which is adopted by IPv4 but not by IPv6, which uses the newer Neighbor Discovery[11] protocol.

The example shown in Figure 2-4 explains the role of the two types of addresses. Suppose that we want to transmit a packet from the host B to the host A. The transmission occurs in the following four phases, through three different packets identified with (a), (b), and (c) in Figure 2-4:

1. The host B generates an IPv6 packet with destination address equal to A and source address equal to B; this packet will remain unchanged until it reaches the destination. B checks whether A is on the same LAN, and if this is not true, B sends the message to R2 by inserting the IPv6 packet into a layer 2 envelope with a destination link address equal to R2 and source link address equal to B (packet (a)).
2. The router R2 receives the packet (a) and uses its routing table to decide to retransmit the packet on the point-to-point WAN link. In this case, as we are in the presence of a point-to-point channel, the presence of link layer addresses in the packet (b) is not necessary.

Figure 2-4
Link and IPv6 addresses



56982_CH02I 12/12/97 3:29 PM Page 30

3. The router R1 receives the packet (b) and decides to retransmit it to A through the LAN. By using the Neighbor Discovery algorithm, it discovers the link layer address of A starting from its IPv6 layer address and then executes the transmission of the packet (c).
4. The host A receives the packet (c) and, because the IPv6 destination address is equal to its layer 3 address, it doesn't send the packet further in the network but passes it to its upper layers.

2.7 Neighbor Discovery

To manage the interaction between different nodes connected to the same link (for example, to the same LAN), IPv6 uses ICMP (Internet Control Message Protocol)[11, 12] messages.

These messages have the following three purposes:

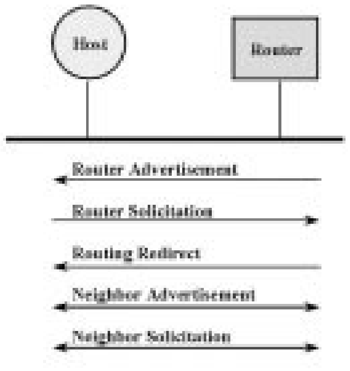
- To allow hosts to know which routers are present on a link. This capability is implemented through periodical multicast transmission of the ICMP *Router Advertisement* packet. Router Advertisement messages are transmitted by routers and received by all the hosts connected to a link that stores, in this way, the presence of routers in a local cache.
- To allow hosts to learn through *Routing Redirect* packets which is the best router through which a node outside the link can be reached.
- To allow all nodes (hosts and routers) to learn mappings between IPv6 addresses and link addresses through *Neighbor Solicitation* and *Neighbor Advertisement* messages.

Figure 2-5 shows the five types of packets and their direction.

2.7.1 Router Advertisement

Routers use Router Advertisement messages to advertise their presence on all links to which they are connected. This process can happen periodically or as a response to a *Router Solicitation* message. Router Advertisement messages contain several parameters relevant to the link, among which are addresses, prefixes, and so on.

Figure 2-5
Neighbor discovery messages



These types of messages allow hosts to learn all routers present on a given link automatically, and they overcome one of the main IPv4 limits: the manual configuration of a default router.

Router Advertisement messages are used by hosts to build their *Default Router List* automatically.

2.7.2 Router Solicitation

When the interface of a host becomes active, it can send a Router Solicitation message to request all routers connected to the link to send a Router Advertisement message immediately, without waiting for the periodical transmission.

2.7.3 Routing Redirect

When a host must communicate for the first time with a destination on a subnetwork to which the host is not directly connected, it must choose a default router from its Default Router List and send the packet to it. The chosen router cannot represent the best choice and be forced to route the packet toward another router on the same link from which it received the packet. In this case, the chosen router, besides correctly delivering the packet, generates a Routing Redirect message to signal to the host that there is, on the same link, a router that represents a best choice toward the final destination.

The host, when receiving a Routing Redirect message, updates its *Destination Cache*, storing the best path.

2.7.4 Neighbor Solicitation

A Neighbor Solicitation message is sent by a node to discover the link layer address of another node or to check whether another node is still reachable through the address stored in the cache. This message is also used in the autoconfiguration phase to detect the presence of duplicated addresses.

2.7.5 Neighbor Advertisement

A Neighbor Advertisement represents the response to a Neighbor Solicitation message. A node can periodically send this type of message as well. When a node receives this type of message, it updates its *Neighbor Cache*, which contains the mapping between IPv6 and layer 2 addresses.

The Neighbor Advertisement message, with the Neighbor Solicitation message, replaces the IPv4 ARP[10] protocol.

2.8 Encapsulation of IPv6 on LANs

IPv6 must coexist on LANs with many other protocols, one of which is IPv4. For a long time, IPv6 designers discussed how to implement this coexistence, by mainly analyzing the following two options:

1. To consider IPv6 as an evolution of IPv4 and therefore to maintain, at the local network level, the *Protocol Type* equal to that of IPv4 (that is, 0800 hexadecimal). This solution entails IPv4 and IPv6 packets being distinguished by the *Version* field (that is, by the first four bits of the IP packet). (See Figures 1-3 and 1-4.)
2. To consider IPv6 as a new protocol completely different from IPv4 and therefore to assign a *Protocol Type* different from that of IPv4.

The latter solution was chosen because it is more robust and reliable during the migration from IPv4 to IPv6, when both protocols will be active at the same time. The new assigned Protocol Type, 86DD (hexadecimal), and the LAN encapsulation are shown in Figure 2-6.

The solution (b) can be used on all IEEE 802 (IEEE 802.3, 802.5, FDDI, and so on) LANs; it anticipates that after the MAC header (MAC-DSAP, MAC-SSAP, and Length), the LAN LLC header will be present in its SNAP

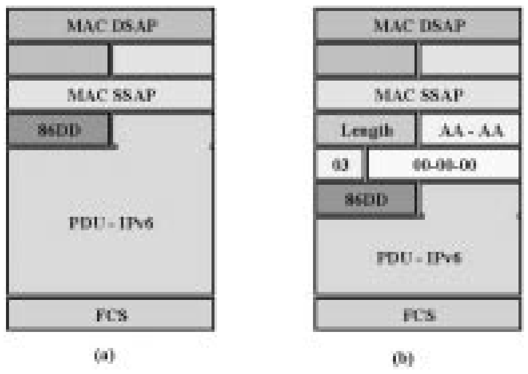


Figure 2-6 Encapsulations of IPv6 on LANs

variant (see Chapter 5 of *Reti Locali: dal Cablaggio all'Internetworking*[4]). The solution (a) is used only on Ethernet and IEEE 802.3 LANs, but it is very important because of the wide diffusion of this type of network.

2.9 Impact of IPv6 on Upper Layers

The TCP/IP network architecture is not perfectly layered; therefore, the replacement of the IPv4 protocol with the IPv6 protocol has an impact also on upper layers (for example, TCP and UDP) up to involved applications (for example, Telnet, FTP, SMTP).

The first aspect to be considered is that applications allow us to specify the destination node by using its IP address or its name. In the latter case, applications use the Domain Name Service to map the name into the corresponding address.

In both cases, they must be modified to manage new IPv6 addresses on 128 bits.

These addresses are typically passed to TCP and UDP transport protocols, which must be updated, too. In the case of TCP (Transmission Control Protocol)[13], modifications are even more substantial. In fact, TCP also uses source and destination IP addresses as connection identifiers; therefore, its data structures must be updated.

In general, enabling TCP and UDP to work is necessary either if the network layer is IPv4 or if it is IPv6. In fact, we can realistically think that, during the transition period, many hosts will support both IPv4 and IPv6 at the same time.

2.10 Modifications to Sockets

To update all applications, even those written by end users and not only those belonging to operating systems, redefining sockets so that they are both IPv4 and IPv6 compatible is necessary.

To accomplish this task, *Basic Socket Interface Extensions for IPv6*[14] supplies new definitions to be used with operating systems derived from Berkeley UNIX (4.x BSD); these definitions can be implemented on all other operating systems.

2.10.1 New Macro Definition

First, a new macro called **`AF_INET6`** has been defined in **`<sys/socket.h>`** with the purpose of differentiating the original data structure **`sockaddr_in`** from the new data structure **`sockaddr_in6`**. In parallel, a new macro called **`PF_INET6`** (Protocol Family) has been defined, and its value is set equal to **`AF_INET6`**.

2.10.2 Definition of the Data Structure for IPv6 Addresses

The data structure that will contain an IPv6 address has been defined in the file **`<netinet/in.h>`** in the following way:

```
struct in6_addr {
        u_char  s6_addr[16];  /* IPv6 address */
}
```

This data structure contains a set of 16 elements, each 8 bits long, unsigned, that together form the 128-bit IPv6 address.

The structure **`in6_addr`** is used to build the new structure **`sockaddr_in6`**, which is used to contain the address of a socket and is defined in the following way:

```
struct sockaddr_in6 {
   u_short   sin6_family;    /* AF_INET6 */
   u_short   sin6_port; /* Transport layer port # */
   u_long    sin6_flowinfo;/* IPv6 flow information */
```

```
    struct in6_addr sin6_addr;      /* IPv6 address */
};
```

2.10.3 The socket( ) Function

Application programs use the **socket()** function to create a socket descriptor that represents the endpoint of a communication. Parameters passed to the **socket()** function indicate which protocol must be used and which is the address's format. For example, to create a TCP connection on IPv4, a call of the following type is used:

```
s = socket (PF_INET, SOCK_STREAM, 0);
```

The value **PF_INET** is used as the first parameter of the **socket()** function to request the creation of a socket on IPv4. If we want to create the same connection but use IPv6, we need to specify **PF_INET6** as the first parameter:

```
s = socket (PF_INET6, SOCK_STREAM, 0);
```

2.10.4 Interoperability

To guarantee the usability of all current applications, the new API (Application Programming Interface) must be compatible with the old one either at the source level or at the binary level. This means that an old application can continue to create TCP and UDP sockets on IPv4 by specifying the **PF_INET** parameter in the **socket()** function. In general, creating any combination of TCP and UDP communications on IPv4 and IPv6 also within the same process must be possible.

2.10.5 Mapping Names into Addresses and Vice Versa

To map names into addresses and vice versa, the decision was to adopt what was defined by the standard POSIX 1003.1g (Protocol Independent Interfaces)[15]—that is, **getaddrinfo()** functions (for mapping names into addresses) and **getnameinfo()** functions (for mapping addresses into names). These two functions were designed by IEEE to be independent from the protocol and are therefore suitable to meet IPv6 needs.

2.10.6 Mapping Binary Addresses into ASCII Addresses and Vice Versa

Each time we need to interact with human users, we need to translate an address's numerical format into a textual format or vice versa. To do so, we can use the two new library functions that have been defined:

`inet_pton()` (from a textual format to numerical a format)

`inet_ntop()` (from a numerical format to a textual format)

2.11 Domain Name Service (DNS) Modifications

The calls to functions **`getaddrinfo()`** and **`getnameinfo()`** cannot be executed if the Domain Name Service is not upgraded, allowing it to store IPv6 addresses.

First, a new type of record "AAAA"[16] has been added. The name of this new record (AAAA) was derived from the one used to memorize IPv4 (A) addresses; because IPv6 addresses are four times bigger than IPv4 addresses (128 bits instead of 32), the decision was to use four *A*'s.

Therefore, if, in DNS, we write configuration files mapping from the name into the IPv4 address as

```
HOST1.POLITO.IT IN A 130.192.253.252
```

we write the same operation from the name into the IPv6 address as

```
HOST1.POLITO.IT IN AAAA 4321:0:1:2:3:4:567:89ab
```

The DNS must also provide opposite definitions—that is, of mapping addresses into names. To define the mapping from an IPv4 address into a name, we use a PTR record, for example, with reference to the previous case:

```
252.253.192.130.IN-ADDR.ARPA. PTR HOST1.POLITO.IT
```

Because the ARPA domain is obsolete, it has been decided to define the second layer IP6 domain under the first layer INT domain. With reference to the preceding example, the rule to map an IPv6 address into the corresponding name is as follows:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.
      3.4.IP6.INT. PTR HOST1.POLITO.IT
```

2.12 DHCP Servers

In practice, the length of IPv6 addresses makes their use by end users impossible. End users will work on IPv6 using only names, and these names will be converted into addresses by DNSs. Also, network managers will be confronted with the addresses' lengths, so they must adopt the necessary support tools for the network configuration. In particular, configuring IPv6 addresses not directly on hosts, but on DHCP (Dynamic Host Configuration Protocol)[17] servers, will become common. Hosts, when bootstrapping, will interact with DHCP servers to configure their addresses and their prefixes (the subnetworks).

In practice, DHCP servers are databases that contain relationships between link addresses (typically LANs' MAC addresses) and IPv6 addresses, whereas DNS servers contain relationships between IPv6 addresses and names. Because both types of servers (DNS and DHCP) will be practically mandatory with IPv6 and because both of them share IPv6 addresses, integrated solutions for DHCP and DNS servers based on a common database should be preferred.

REFERENCES

[1]J. Postel, *RFC 791: Internet Protocol*, September 1981.

[2]IS 8473, *Information processing systems—Data communications—Protocol for providing the connectionless-mode network service*, ISO, 1988.

[3]S. Deering, R. Hinden, *RFC 1883: Internet Protocol, Version 6 (IPv6) Specification*, December 1995.

[4]S. Gai, P.L. Montessoro, P. Nicoletti, *Reti Locali: dal Cablaggio all'Internetworking*, SSGRR (Scuola Superiore G. Reiss Romoli), 1995.

[5]R. Hinden, S. Deering, *RFC 1884: IP Version 6 Addressing Architecture*, December 1995.

[6]S. Thomson, C. Huitema, *RFC 1886: DNS Extensions to support IP version 6*, December 1995.

[7]G. Bennett, *Designing TCP/IP Internetworks*, Van Nostrand Reinhold.

[8]G. Malkin, *RFC 1723: RIP Version 2—Carrying Additional Information*, November 1994.

[9]J. Moy, *RFC 1583: OSPF Version 2*, March 1994.

[10]D.C. Plummer, *RFC 826: Ethernet Address Resolution Protocol: On converting network protocol addresses to 48 bit Ethernet address for transmission on Ethernet hardware*, November 1982.

[11]T. Narten, E. Nordmark, W. Simpson, *RFC 1970: Neighbor Discovery for IP Version 6 (IPv6)*, August 1996.

[12]A. Conta, S. Deering, *RFC 1885: Internet Control Message Protocol (ICMPv6)*, December 1995.

[13]J. Postel, *RFC 793: Transmission Control Protocol*, September 1981.

[14]R.E. Gilligan, S. Thomson, J. Bound, *Basic Socket Interface Extensions for IPv6*, IETF, April 1996.

[15]IEEE, *Protocol Independent Interfaces*, IEEE Std 1003.1g, DRAFT 6.3., November 1995.

[16]S. Thomson, C. Huitema, *RFC 1886: DNS Extensions to support IP version 6*, December 1995.

[17]R. Droms, *RFC 1541: Dynamic Host Configuration Protocol*, October 1993.