Domains and IPv6

Jan-Erik Ekberg
Carl Eklund

October 26, 1998

Contents

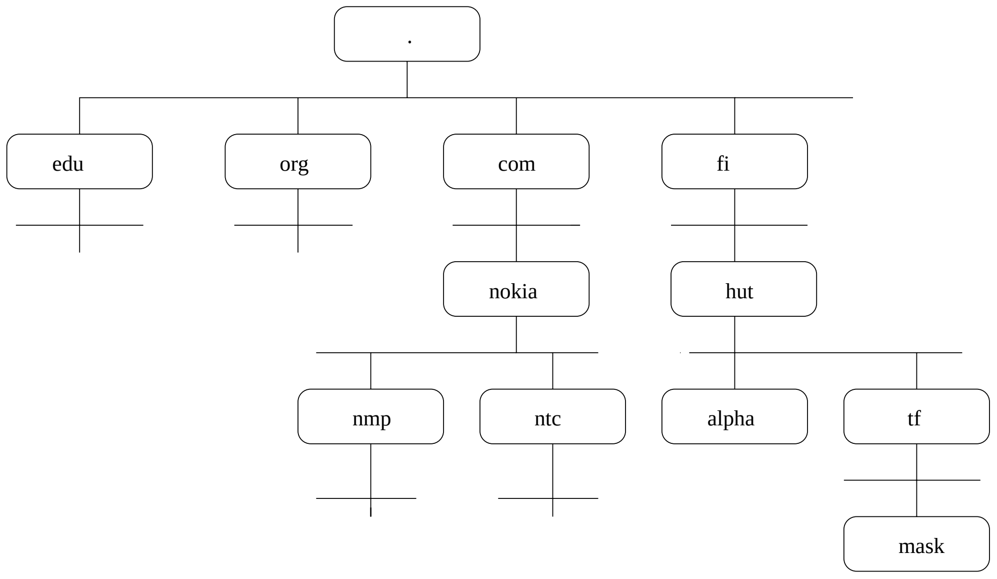


Figure 1: The domain tree

Part I
Domains

1 Domain Names

1.1 Name hierarchy

An Internet name consists of a series of labels separated by dots. The number of labels is not limited but each label cannot exceed 63 characters. The total length of the name is limited to 255 characters. The name can contain the letters A through Z in upper case, the letters a through z in lower case, the digits 0-9 and the hyphen '-'. The names are case insensitive.

Internet names have a tree structure, as shown in figure 1. Each node in the tree corresponds to a label. Each node is referred to by its *domain name*. It is made up from the sequence of labels from the node to the root of the tree. The node's domain name is written as the sequence of labels separated by dots.

A *domain* is defined as the part of the naming tree that consists of a node and all the nodes below it.

The label for the root is a period. Thus every Internet name ends in a period. The final period is usually omitted when citing the name, e.g. `gamma.hut.fi.` is written as `gamma.hut.fi` in most cases[1, 2]. The *top-level domains* below it reflect the history of the Internet and are:

- `edu`, universities and degree granting institutions in the US
- `gov`, US federal government agencies
- `com`, commercial organizations
- `net`, Internet network service organizations

- `org`, non-profit organizations
- `int`, international organizations
- `mil`, US military organizations
- Countries, two character ISO country codes. The structure of the tree under the country code is delegated to local administrators.

1.2 Administration of domain names

The Internet names and addresses were initially administered by the United States Department of Defense Network Information Center DDN NIC. In 1993 the responsibility for non-military names and addresses was given to the US National Science Foundation. The NSF funds the InterNIC Registration Service which is the primary authority for worldwide naming and addressing. InterNIC has delegated authority to two major regional registries, the Asia Pacific NIC and the RIPE[1] Network Coordination Center (for Europe). The InterNIC and the regional registries in turn delegate responsibility to national and local registries within their region[3]. In Finland the registry is administered by the Telecommunications Administration Center. The domain in question is the `fi.` domain.

The procedure for registering a domain name in Finland is specified in document THK 34/1997M. It also lists a number of requirements that domain names must fulfill. Any registered company, foundation or society can apply for a domain name within the `fi.` domain. Usually one domain name is granted per qualified applicant, but several names can be given for special reasons. The name should be clear and associatable to the name owner. The name must not unreasonably limit the possibilities of other eligible societies to get a domain name. Also, it must not offend the the legal right to a registered company name or trademark etc. and may not be misleading or offending. The TAC also specifies the minimum technology requirements for a domain name holders network in document THK 37/1997. The requirements are that there are two independent name servers for the domain and that the mail service is properly implemented[4, 5].

The non-national top-level domains .com, .net and .org are administered by a commercial company, Network Solutions. The situation has been controversial. One reason for this is the granting of .com addresses. European parties have feared that Network Solutions would side with US companies in disputes over certain domain names. As a solution there is a proposal to for a new private international non-profit corporation, Internet Corporation for Assigned Names and Numbers (ICANN) that would replace Network Solutions and the Internet Assigned Numbers Authority (IANA)[6]. At the same time a development has taken place that potentially decreases the commercial importance of domain names. Netscape's newest browser incorporates keyword browsing, that allows browsing without the knowledge of the URLs. Microsoft will also include a similar functionality in its browsers[7, 8].

[1]Reseaux IP Europeens

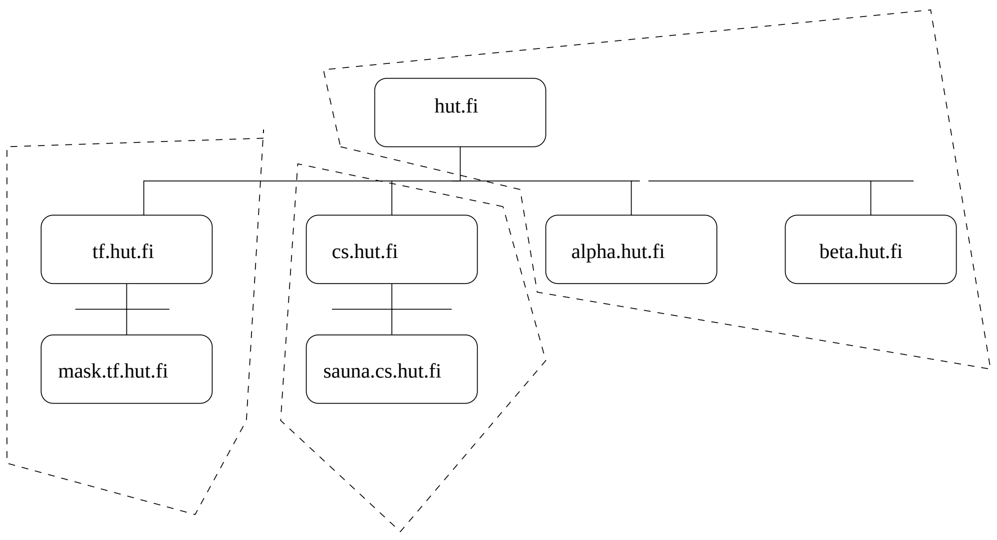


Figure 2: Division of a domain into autonomous zones

2 Domain Name System

2.1 Introduction

Most users know the name of the hosts the are utilizing but are not concerned with its address. IP, howerer needs to know the addresses in order to work. This means that an name to address conversion has to be made. In the early days of the Internet all host names were kept in a central table host name to address conversion table, that was distributed to individual networks periodically. This approach soon became impractical. The Domain Name System(DNS) was introduced to provide a better method of managing Internet names and addresses[3].

2.2 DNS

DNS is a distributed database. Internet names and addresses are kept at servers run by the organizations that owns a domain name. The mail servers also contain the mail routing information for the organization. The domain name owner is responsible for running and maintaining the name servers that translate the host names of the domain to their corresponding addresses. Thus the changes to a local network can quickly be entered at the domain's primary server. As DNS is a critical service for the Internet the information has to be replicated at one or more secondary servers. The servers should furthermore be a independent as possible i.e. preferably be located in different physical networks.

The naming tree of an organization consists of one or more *zones*. A zone is a contiguous part of the naming tree that us administered as a unit. A part of the zone structure for the hut.fi domain is shown in figure 2. The root name servers for the fi domain would provide a pointer to the hut.fi name server. It is the primary name server for the hut.fi domain with exception to the autonomous zones. The cs.hut.fi and tf.hut.fi zones have their own primary name servers into which changes in the respective zones are entered. It would also be acceptable to use one server for for multiple zones or even domains. The data for each zone is stored separately.

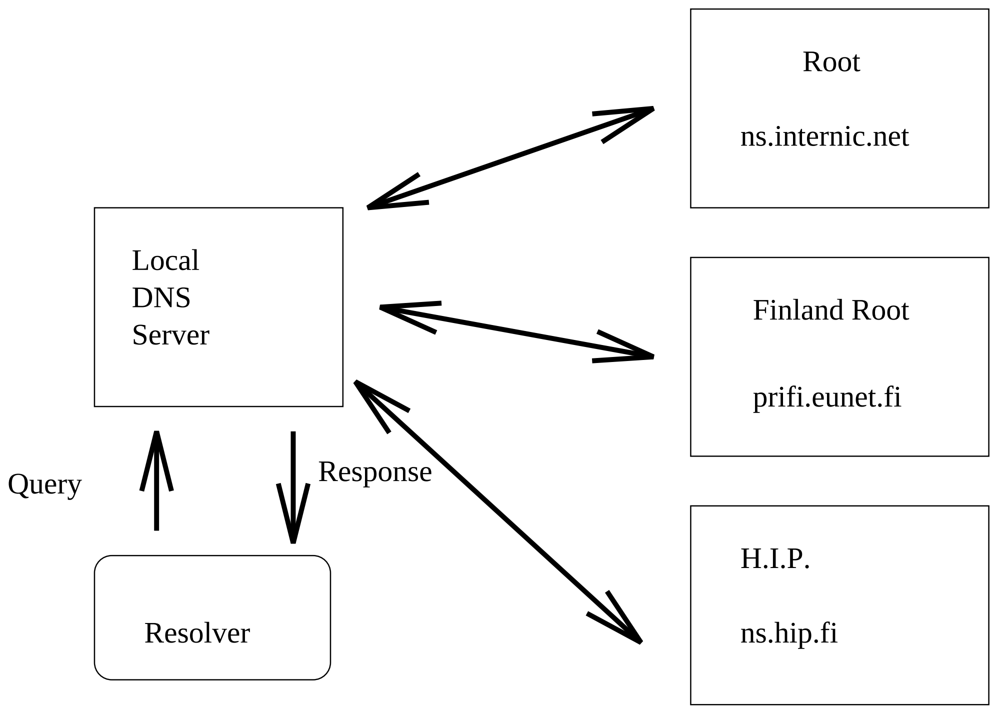


Figure 3: Example name lookup

Secondary servers are set up to provide access to a copy of the information about one or more zones. They receive the info from the primary servers via zone transfers. A secondary server can mirror several zones from different primary servers. It is also possible for a server to act as a primary server for certain zones and a secondary for others[1, 2].

2.3 Name-to-address resolution

A client program capable of looking up information in the DNS is called a *resolver* and is a standard part of TCP/IP products. The resolver is usually invoked by an application and is transparent to the user. When TCP/IP is installed on a host that intends to use DNS information the must be configured with the IP addresses of two or more Domain Name Servers in order for the resolver to know where to start its query.

In a stand-alone network all the queries an be answered by the local name servers. If, however, the network of an organization is connected to the Internet the name servers will have to retrieve global information. In this case the names and addresses of the Domain Name Servers of the organization have to be added by the appropriate registration authority to its root list of Domain Name Servers. The local name server also need to be configured with the list of the root servers.

The name to address resolution might take place as follows: The resolver client queries the local nameserver for an address corresponding to a name. The local nameserver checks if the name belongs to the local domain and if this is the case it provides the requested information. If the name queried belong to an external host then the nameserver checks whether the info can be found in its cache. The DNS server either replies to the resolver or queries a root server. The root server replies with the names and addresses of the name server providing

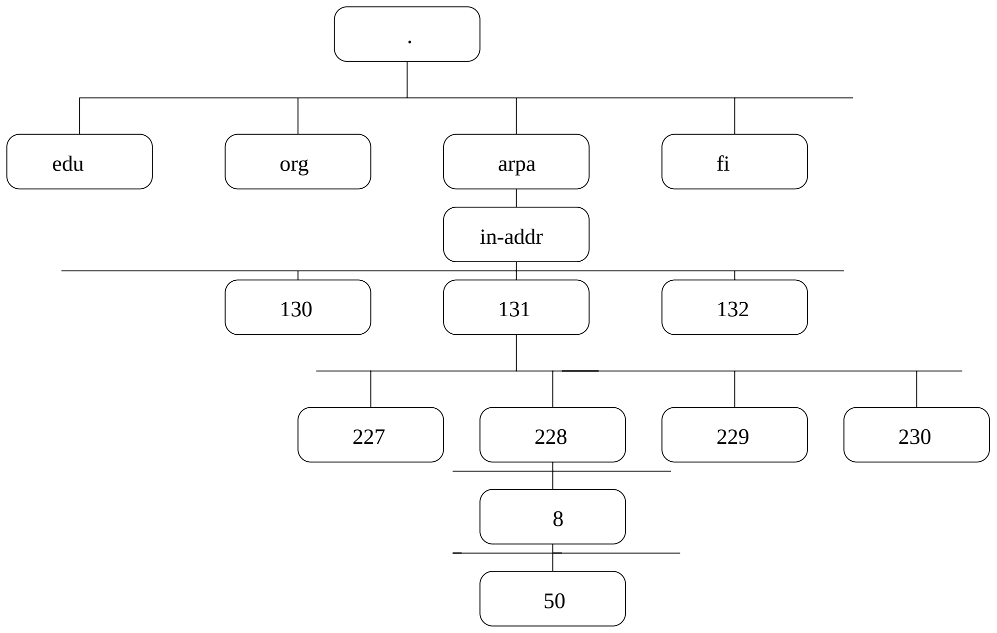


Figure 4: The `in-addr.arpa` domain

DNS service for the top level domain in question. The local nameserver caches the information and then queries one of these servers for the requested host address. This process continues until the desired information is found or an authoritative negative answer was received. The server then finally provides the answer to the resolver. Shown in figure 3 is a query for an address in the `hip.fi` domain. As in this case the resolver usually asks for a recursive resolution. The nameserver, however, usually performs its task iteratively. The DNS queries and responses are normally transmitted via UDP, but TCP is also permitted.

The DNS can also be used to do address-to-name translations. The organization that owns a network address is responsible for recording its address-to-name translations in the DNS database. All Internet addresses have been put into a special domain called *in-addr.arpa*. The structure of the domain is shown in figure 4. Since the most general part of the address must be higher up in the domain tree the order of the numbers in each address is reversed i.e. the subtree for the network 172.66 is called 66.172.in-addr.arpa. Entries in the DNS database are also reverse[1, 2].

2.4 DNS Entries and Resource Records

DNS data is stored as text entries. Each entry is of the form:

```
[name ][TTL][class ]Record-Type Record-Data [;comments ]
```

The time-to-live TTL gives the maximum time the record may be cached after retrieval. The class value currently used is 'IN' for Interet. The name and class values default to the previous if omitted. The order of class and TTL fields may be reversed as there is no risk for confusion. The part of the data entry consiting of

```
[TTL][class ]Record-Type Record-Data
```

is called a *Resource Record(RR).* The RRs are identified by characters or short acronyms. The most important RRs are:

- SOA, identifies the domain or zone. Sets parameters and contact information for administrator.
- NS, maps a domain name to the name of a computer that is authoritative for the domain.
- A, maps a system name to its address. For a host with several notwork interfaces there has to one record per interface.
- CNAME, maps an alias name to the true name i.e. the *canonical name.*
- MX, Mail Exchanger.
- PTR, maps an IP address to a system name.

2.5 DNS security extensions

The DNS security extensions provide services for key distribution, data origin authentication, and transaction and request authentication.

For key distribution a RR, KEY, has been defined to associate keys with DNS names. The RR fields also specify the algorithm, the type of key association.

Data origin authentication and integrity checking is provided by assocating with the RRs a digital signature. Usually there will be a single private key that signs for an entire zone. Once a security aware resolver learns the public key of the zone it can verify the signed data. The SIG RR has been defined for this purpose. [9]

2.6 DNS dynamic updates

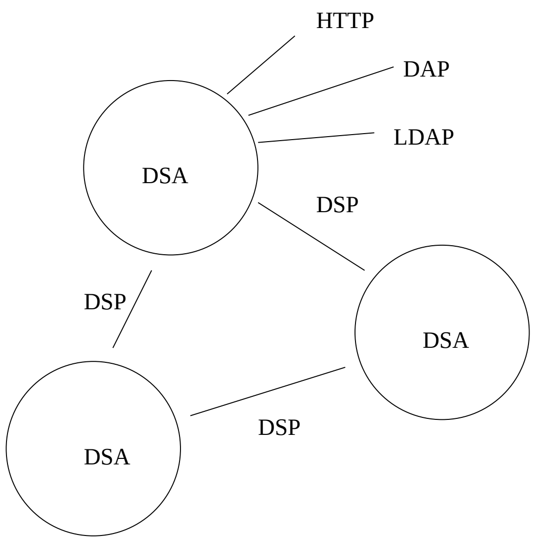
The DNS was originally designed to support queries of a statically configured database. While the data was assumed to change the frequency of change was expected to be low. Thus all updates were made as external updates to the zone's master file. To make possible automatic updates e.g. for a mobile IP node, a new RR, UPDATE was introduced. Using UPDATE is possible to add or remove RRs from a specified zone. Prerequisites for updates are separately specified. Thus an update can be specified to depend on the the existence or nonexistence of previous RRs. The dynamic updates also support DNS security extensions[10, 11].

3 LDAP

3.1 Introduction

The LDAP protocol is a directory access protocol, originally drafted to provide a simple means of access to the global X.500 directory structure, see fig.5. The information stored in the X.500 typically consists of addresses and identifiers of different kinds: mail addresses, public certificates for authentication, current customer location information, customer care data etc.

The main simplification carried out in LDAP, compared to the X.500, is that attributes are referred to by simple data types such as strings and integers, instead of the elaborate



DSP	Directory System Protocol
DSA	Directory System Agent
DAP	Directory Access Protocol
LDAP	Lightweight DAP
HTTP	HyperText Transfer Protocol

Figure 5: LDAP in relation to X500

ASN.1 - data types in use within the X.500 - structure. Also, the search possibilities of the LDAP are somewhat restricted compared to the full search capabilities supported by the X.500.

Traditionally the LDAP protocol has mainly been used in ISO/OSI environments, e.g. for retrieving and sorting out X.400 mail addresses, or in some network security solution for accessing X.509 public key certificates. Within the internet, the LDAP protocol has been relatively unheard of. However, a growing interest towards the protocol and its generality can be seen e.g. in the Internet Telephony field, where access to customer (location) information are retrievable by LDAP (Microsoft ILS, Netscape DLS), or in Microsoft's recent promises to base its network information services on LDAP in the forthcoming Windows NT 5.0 release. Also some ISP billing software uses LDAP for customer information storage.

The current version of the LDAP protocol is v.3. The main new feature is that the client part of the protocol may be presented with referrals (requests to continue the search at some other server), although there are other minor feature updates in the actual protocol and its message structures as well.

There are many good introductory and index pages for LDAP implementations and documents on the internet. A good place to start is [12] or [13] for a more business oriented view of the protocol.

3.2 The network layer of LDAP

Starting from the 'bottom', the LDAP protocol uses its own TCP-connections for communication. A client opens a TCP/IP connection to the server, and 'binds' to that connection. Is is understood, that the server will keep the connection until the client 'unbinds' - i.e several database queries can take place during one TCP/IP connection (in this respect it differs from e.g. the HTTP protocol). The transmitted messages are structured according to ASN.1, and coded in the Distinguished Encoding format (DER). In its basic form the protocol consists of search requests (or updates) from the clients, and various responses from the server.

The network messages are in principle authenticated using the 'Simple Authentication and Security Layer' for connection-oriented protocols, defined in [14]. Its most basic form is to transmit passwords in plaintext during binding of the session, but the 'correct' usage of the protocol is more elaborate. The basic idea is to send an authentication / authorization method descriptor in place of a user-name / password pair, which on the server and client side will trigger a separate message exchange between the peers, or as the protocol summary states:

> "If a server supports the requested mechanism, it initiates an authentication protocol exchange. This consists of a series of server challenges and client responses that are specific to the requested mechanism. The challenges and responses are defined by the mechanisms as binary tokens of arbitrary length. The protocol's profile then specifies how these binary tokens are then encoded for transfer over the connection."

...

> "If use of a security layer is negotiated, it is applied to all subsequent data sent over the connection. The security layer takes effect immediately following the last response of the authentication exchange for data sent by the client and the completion indication for data sent by the server."

Unfortunately, there was no consensus of mandatory authentication and confidentiality protocol when the LDAPv3 was released - instead a comment stating that "implementors are discouraged from deploying LDAPv3 clients or servers with update functionality until a Proposed Standard for mandatory authentication in LDAPv3 has been approved and published as an RFC".

3.2.1 LDAP messages

The basic functionality of the LDAP server, is naturally to respond to searches made by clients. The 'Search Request' message has the following outline:

From fig.6 we note, that a client must restrict its searches to one of three different forms, namely the found base objects themselves, the base objects augmented with one sublevel, or the complete subtrees of the found base objects. The filter-parameter contains the query string that may include operators such as AND, OR, NOT, LESS THAN, EQUAL, PRESENT or APPROXIMATE MATCH, and the attributes-parameter contains a list of the attribute types that are returned for each matching object. The server responds with a reply (fig.7), containing the returned attributes in the matchedDN field (depending on the resultCode). Also a referral parameter may be present, it may contain one or several URLs to other LDAP servers known to contain the information that the user requested.

3.2.2 Shortly about the X.500 naming convention and schemas:

In principle the LDAP directory can contain any kind of information, or - in X.500 terminology - conform to any directory schema. The schema of a directory is a set of descriptors (attribute

```
 SearchRequest ::= [APPLICATION 3] SEQUENCE {
         baseObject      LDAPDN,
         scope           ENUMERATED {
                 baseObject              (0),
                 singleLevel             (1),
                 wholeSubtree            (2) },
         derefAliases    ENUMERATED {
                 neverDerefAliases       (0),
                 derefInSearching        (1),
                 derefFindingBaseObj     (2),
                 derefAlways             (3) },
         sizeLimit       INTEGER (0 .. maxInt),
         timeLimit       INTEGER (0 .. maxInt),
         typesOnly       BOOLEAN,
         filter          Filter,
         attributes      AttributeDescriptionList }
```

Figure 6: LDAP Search request

```
 LDAPResult ::= SEQUENCE {
          resultCode      ENUMERATED {
                       success                      (0),
                       operationsError              (1),
...},
          matchedDN       LDAPDN,
          errorMessage    LDAPString,
          referral        [3] Referral OPTIONAL }
```

Figure 7: LDAP Result message

```
5.8. l

   This attribute contains the name of a locality, such as a city,
   county or other geographic region (localityName).

    ( 2.5.4.7 NAME 'l' SUP name )
```

Figure 8: Example of an attribute from [15]

```
7.11. residentialPerson

   ( 2.5.6.10 NAME 'residentialPerson' SUP person STRUCTURAL MUST l
     MAY ( businessCategory $ x121Address $ registeredAddress $
     destinationIndicator $ preferredDeliveryMethod $ telexNumber $
     teletexTerminalIdentifier $ telephoneNumber $
     internationaliSDNNumber $
     facsimileTelephoneNumber $ preferredDeliveryMethod $ street $
     postOfficeBox $ postalCode $ postalAddress $
     physicalDeliveryOfficeName $ st $ l ) )
```

Figure 9: Example of an Object Class from [15]

definitions) (retrievable from the directory as any other data) defining the data contained within the directory. In practice, as the purpose of LDAP servers is mainly to contain address information, a basic 'User Schema' has been defined in [15], in order to synchronize the data types of the most common information related to addressing of people and other objects. A schema consists of object classes, which are entities built up of mandatory and optional attribute types (e.g. fig.8), pinpointing the address of the object that is defined. Object classes (fig.9) are e.g. organizations, residential persons, certification authorities or countries. As an example, a residential person must be defined by means of a common name, a surname and a location, but he/she may be defined by other optional attribute types such as postal address, a telephone number or a street name. The object classes are structured in a tree, i.e. a residential person is a superset of a person, which is also the case for an organizational person, meaning that in both cases the object must be identified by means of a surname and a common name.

All attribute types as well as object classes are defined by a unique number, an OBJECT IDENTIFIER, that is used to define the format and type of any attribute value. The OBJECT IDENTIFIER structure is an essential component of the X.500 - directory structure, and these identifiers uniquely identify everything from algorithms to naming conventions to string formats.

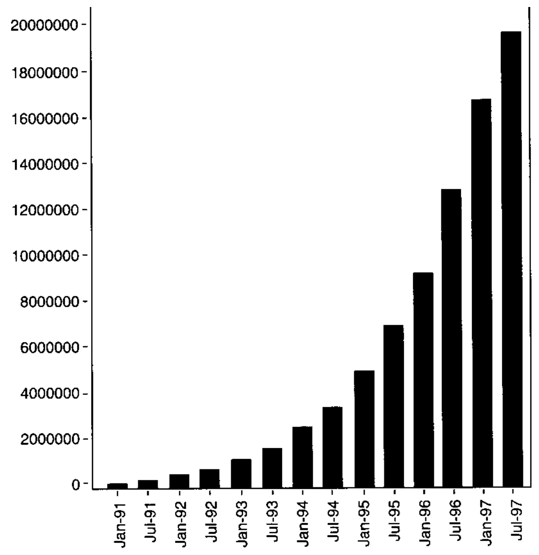


Figure 10: Hosts in the internet (source: http://www.nw.com)

Part II
IPv6

4 IPv6

4.1 Introduction

Even compared to the evolution of the computer industry, the growth of the Internet (and data networking in general) has been truly rapid, and no direct signs show that this trend is going to change anytime soon (see fig.10). During the late 80's and the 90's the evolution of the Internet has made it a connection point for everybody, computer literate or not, and the only real backbone for real services such as e-mail or file transfer. And as everybody surely knows, the address space of the 'old internet', i.e. version 4, is claimed to become depleted, and because of this we are forced to switch to the new version 6 of the protocol.

The address space depletion should - perhaps - not be taken as seriously as it is often is marketed. Today, most machines in the internet are only temporary visitors connecting to the fringe of the network to download information from servers and maybe upload e-mail or similar data. These computers do not necessarily need their own, unique, IP-addresses. The addresses may well be temporarily assigned, or even chosen from a purely local set of addresses if the connection to the internet is routed through an address conversion proxy (commonly called a masquerading machine). So, if permanent addresses were reserved only for those server machines continuously operating in the network, there is room, even with the current 'policy' of address distribution, for hundreds of millions of servers - which seems like a lot.

So, perhaps as important for the decision to upgrade, or maybe more, are the new challenges imposed on the internet protocol - reasons not anticipated in the late 70's when the protocol was designed. The race for terminal mobility has already embraced telephone (speech terminals), and there is no reason why the same wouldn't happen in the datacom world as soon as the technology is mature enough to support it. Also, common people want a certain level of message integrity and confidentiality when e.g. sending and receiving mail or when using services such as home banking or shopping, and this is also a feature that was not designed into the original protocol. The same goes with management support and easy configurability. There are many more reasons to make the switch than just the addressing problem.

There is also a structural reason for upgrading the net. The current solution grew out of an inter- university network, where the maintenance and operation of the network was more or less based on goodwill and research reasons. Now, with the commercialisation of the internet, this approach is difficult to motivate and maintain why should some organizations make money of something that should be open, and free for all. The solution can be seen in the new addressing hierarchy - something of a traditional telephony network structure has been followed, as the address space is divided into a transport-network-part, a service-provider-part and local-subnet-part. The addressing structure will, to a certain level, dictate the routing of a certain packet, making it possible to organize features such as billing and inter-operator clearing better that it is done in the current structure.

Last, a casual observer may conclude that IPv6 is simply hype - there has been talks of it for several years now, but nothing seems to happen. Still, almost all operating systems and network equipment manufacturers are busy implementing IPv6 into their products - the transition must be quick if or when it will happen, but in order to make it so everybody must be ready. So there are clear signs leading us towards the transition, but when and how it will happen nobody knows.

4.2 IPv6 header format

When designing the next version of the IP, the authors claim that they have strived to keep as close to IPv4 as possible (because it works), augmenting only those features that have proved lacking in the previous version, and eliminating largely unused and cumbersome features. This has resulted in a quite clean and simple protocol structure, which is easy to understand and hopefully also to implement.

4.3 The header(s)

The main difference in the IP header in version 6 is, in addition to the enlarged address length, that the extensions (options) have been moved to subheaders (which are comparable to payloads in some sense) resulting in a header with a fixed length of 40 bytes. The contents of the header (fig.11) is:

- The version field (4 bits). This field is in the same place and is of the same length as in version 4. Naturally the value is different.
- A class label (8 bits). This field was originally called priority, which gives one use for the field. The specification of this field is currently left open.
- A flow label (20 bits) defines a stream of data, e.g. voice. A stream is thus uniquely defined by the originating and receiving address in conjunction with the flow label. It is also understood, that intermediate routers could provide some special treatment for

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Prio. |                   Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: IP version 6 header (RFC1883)

packets using this label. The specification of this label is subject to current research, as well,

- The payload length (16 bits), defines the length of the actual payload disregarding all options.
- The next header field (8bits) defines the type of the next data type (after the header). Possible values include TCP, UDP, Authentication Header, Fragment Header or NULL.
- The hop-to-hop count. Note that this is not call time-to-live any more, so the field should actually from this version on contain the hop count - no other metrics are allowed.
- The source add destination address are, as is well known, 128 bits each. The structure of these are described in another section of this document.

Many of the features that are a part of the header in IPv4 have now been moved to the extension headers. Each extension header start with two fields, 8 bits each:

- The option header type. This bit field contains information whether the option changes en route, whether a router should drop or blindly forward a packet where the type of the option is unknown to the router in question. 5 bits are reserved for identifying the actual option type.
- The option header length.

```
+---------------+------------------------
|  IPv6 header  | TCP header + data
|               |
| Next Header = |
|      TCP      |
+---------------+------------------------


+---------------+----------------+------------------------
|  IPv6 header  | Routing header | TCP header + data
|               |                |
| Next Header = |  Next Header = |
|    Routing    |      TCP       |
+---------------+----------------+------------------------


+---------------+----------------+-----------------+-----------------
|  IPv6 header  | Routing header | Fragment header | fragment of TCP
|               |                |                 |  header + data
| Next Header = |  Next Header = |  Next Header =  |
|    Routing    |    Fragment    |       TCP       |
+---------------+----------------+-----------------+-----------------
```

Figure 12: IP extension header chaining (RFC1883)

For all extension headers (defined in the following) there is a padding option defined. In order to support upcoming 64-bit architectures, the header can be configured so that the option fields end on a nice word boundary, and this can be achieved using padding.

The extension headers are chained (see fig.12, so that each extension header defines the type of the next header, and its own length, so that a reasonably simple parser can be constructed. The headers (with the exception of the hop-by-hop header) are processed only at the endpoints and should carry no information that has to be seen by intermediate routers.

There are a number of predefined extension headers, that more or less are considered to be a part of the IPv6 protocol itself. These include:

- The hop-by-hop header. The idea with this extension header is that it should be examined on every hop (and thus it should follow immediately after the IP header itself, if it is present). The specification of this extension header is very much left to the future, and whatever afterthoughts may arrive - currently the only hop-to-hop option that is defined is the Jumbo-payload option for packets that exceed 65535 bytes in size.

- The destination options header. This header should include any parameters that should be examined by the destination host(s) only. Currently the specification has no predefined fields for this extension header.

- The routing header. This is used for source routing in a very traditional manner (note that this header definitely is one that changes it contents along the way). It contains a list of addresses through which the packet should be sent to the destination, and some pointers to identify which leg in the list the packet currently is travelling on.

- The fragment header. Fragmentation differs from that in IPv4 in that fragmentation can only occur end-to-end. Also the need for fragmentation in the new protocol is questionable, as there are methods in IPv6 supporting end-to-end MTU discovery. Anyway, the fragment header contains all necessary information (offset of fragment, more fragments coming).
- The authentication header contains information needed to authenticate the packet. It contains a SPI (Security Parameter Index) which is an end-to-end identifier for a security association. The security association essentially identifies the algorithm that is used to authenticate the packets. There is also space for serial numbers and the actual MAC (Message Authentication Code).
- The encapsulating security payload header supports a higher level of security as it also support confidentiality. It is the last readable extension header, as it will encapsulate all following headers and payloads within itself, encrypted of course. It also contains fields for authentication (similarly to the fields in the authentication header, and naturally an SPI value.

IPv6 defines the minimum allowed MTU for the transmitting network to be 576 bytes. This means that if a network (e.g. ATM) is to be used as a transmission channel, fragmentation and reassembly must be provided at a layer below IPv6 to support this minimum required length. There is no maximum MTU, as big packets can always be defined using the Jumbo Payload option.

4.4 Addressing

The IPv6 protocol defines three basic variation of addresses, which should be more or less familiar to anyone that has stumbled over packet networks before:

- Unicast. A packet sent to a unicast address has a single, uniquely determinable receiver.
- Multicast. A packet sent to a multicast address is delivered to all interfaces / hosts that are identified by that address (or have registered themselves as listeners to that specific address)
- Anycast. This, somewhat esoteric form of addressing, delivers the packet to the nearest interface that is defined by the given address.

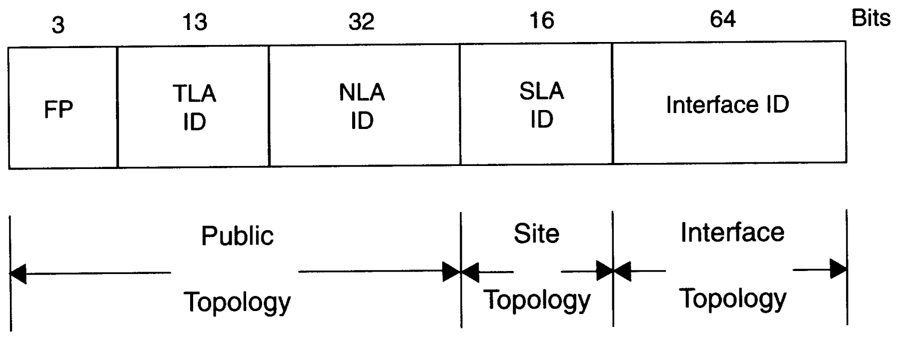
As the address space has grown to 128 bits, a somewhat new way of writing down addresses has been devised. An address is described by X:X:X:X:X:X:X:X, where each X represents 16 bits in hexadecimal representation, e.g. FEDC:BA98:7654:3210:FEDC:BA98:7654:3210. The following special syntax rules are also used:

- Leading zeros in any of the address parts may be omitted, e.g. 185C:0:0:0:0:0:0:0001 = 185C:::::::1

- A double colon may appear in the beginning or at the end of an address, and defines that the beginning or the end of the address is all zeros: 0:0:0:0:0:0:0:1 = ::1
- The address may be followed by a slash (/) and a number, denoting the length of the prefix from the leftmost part of the address, e.g. . FEDC:0:A98:0:0:0:0:0/40 indicates that the prefix is FEDC0000A9

The addressing architecture contains a number of different address types. The most common are presented in the following list:

- The address :: is defined as the unspecified address, an indicates the absence of and address
- The address ::1 is the loopback address for the local host.
- An IPv4 compatible IPv6 address is defined as 96 zero bits, followed by the 32-bit IPv4 address. For IPv4 nodes communicating in an IPv6 network the corresponding address is 80 bits of zeros, followed by 16 bits of ones, followed by the IPv4 address.
- Separate address spaces have been allocated to IPX addresses and OSI NSAP (Network Service Access Point) addresses.
- The unicast address format most likely to be used in a global environment is an Aggregatable Global Unicast Address (see fig.13, which takes its model from the telephone network, in that the address is divided into different segments. The first three bits (001) defines the address type. The following 13 bits (TLA=Top-Level-Aggregation) are allocated to global organizations providing public IPv6 transmission service (these organizations must have a good track record in this business). These 'long-haulers' will allocate the Next-Level Aggregation (NLA) (32 bits) to organizations providing subnets or 'sites' (big organizations, ISPs, etc.). The next 16 bits are allocated to site-topology specification (within the NLA), and the last 64 bits define the interface address of the individual interface. The TLA+NLA define the 'Public Topology' part of the address space.
- Some testing address spaces have also been allocated
- The link-local addresses are valid within one link (ethernet segment) only, and are meant for auto- address configuration etc. The format is '1111111010' + 54 zeros + the interface ID.
- The site-local addresses are restricted to in-site traffic only. The format is '1111111010' + 38 zeros + the subnet ID + the interface ID.
- The multicast addresses are prefixed with '11111111', and continue with a four-bit flag describing whether this multicast address is well-known or non-permanently assigned. The following four bits define the scope (from link-local to global). The final 112 bits defines the group ID (or channel)



Notes:

FP:	Format Prefix (001)
TLA:	Top-Level Aggregation Identifier
NLA:	Next-Level Aggregation Identifier
SLA:	Site-Level Aggregation Identifier
Interface ID:	Interface Identifier

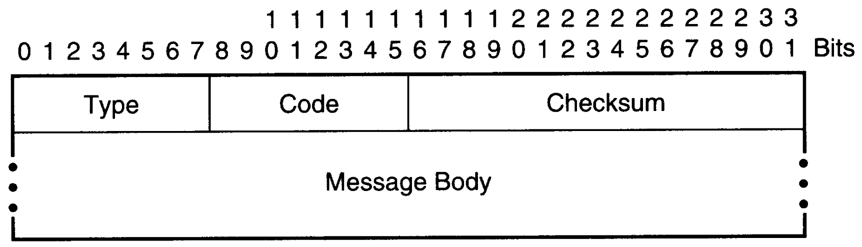
Figure 13: Proposed aggregatable global unicast address ([16])

4.5 ICMP

Whereas ICMP in IPv4 was quite non-homogenous, the ICMP messages in v6 have been streamlined, and have a common structure (fig.14). They are grouped into two categories - error messages and informational messages. The ICMPv6 is perhaps more clearly just one type of payload in the IP packets.

All ICMP packets contain the type of the packet (8 bits) and a codeword, which adds an extra level of specification for certain message types (also 8 bits). A 16 bit checksum verifies that the message is transmitted correctly. The most important messages are as follows:

- Destination Unreachable: The codeword separates between 'No route to destination', 'Communication with destination administratively prohibited', 'Not a neighbor', 'Address unreachable' and 'Port unreachable'. The payload will contain as much of the original message as will fit in 576 octets.
- Packet Too Big: This message contains the maximum MTU of the link for which the resending failed, and as much of the original message as will fit in 576 octets.
- Time Exceeded: Two codewords are defined - hop count exceeded and fragment reassembly time exceeded.
- Parameter Problem: The possible codewords are 'Erroneous header field encountered', 'Unrecognized next header type encountered' and 'Unrecognized IPv6 option encountered'. A pointer identifies the position of the problem (in octets from the beginning of the packet).
- Echo Request / Reply: This ICMP command fulfills the UNIX ping command. Two 16-bit parameters are included - a sequence number and an identifier.



Type Field Value	Meaning
0-127	ICMPv6 error messages
1	Destination unreachable
2	Packet too big
3	Time exceeded
4	Parameter problem
128-255	ICMPv6 informational messages
128	Echo request
129	Echo reply
130	Group membership query
131	Group membership report
132	Group membership reduction
133	Router solicitation
134	Router advertisement
135	Neighbor solicitation
136	Neighbor advertisement
137	Redirect

Figure 14: ICMP messages ([16])

Two additional sets of ICMP messages define group membership and neighbor discovery. The first set consists of three messages which are quite equivalent to the correspondent IPv4 protocol messages - a membership query and membership reply and reduction messages (all include a maximum response delay - parameter). The idea behind these messages are as follows.

4.5.1 Group membership messages

A node wishing to know what other hosts in the local subnet belong to a specific multicast group may send a group membership query to that multicast address. All hosts that are in the group will respond with a group membership report, again using the multicast address, meaning that all interested hosts will be able to update their group membership tables at the same time. A node wishing to leave a multicast group may do so with a group membership reduction message, so that all nodes in the group may update their membership tables accordingly.

The group management messages could be used to build the multicast overlay network (MBONEv6) for IPv6, but as this field is still under study any comments hit on a moving target. According to [17], some changes that are expected in the coming years are:

- One or several source addresses in the group membership report could indicate that a client wishes to listen to a specific subset of sources within the group.
- Similarily, the reduction message could state which nodes a terminal doesn't want to listen to.
- Some multicast routing protocols need to ascertain which of the link's routers will relay packets for a specific group. Some additional signalling may be needed to support this functionality.

4.5.2 Neighbor discovery

The second set of messages is related to the concept of neighbor discovery, which is a much advertised concept in IPv6. The following problems can be solved using these commands:

- Router and subnet prefix discovery. How does a node find a suitable gateway out of the local subnet for a specific packet, and what is the global address of the subnet.
- Parameter discovery: Some IPv6 parameters should be set in a dynamical fashion - these include the hop count and the MTU.
- Address autoconfiguration and Address resolution. The first one is discussed in greater detail in the section about IPv6 mobility, the second one is used (as in IPv4) to logically connect interface link addresses to IP addresses (the so-called ARP-protocol).

The messages are roughly as follows:

- Router solicitation and router advertisements. These multicasted messages form the basic means for a client to find suitable routers to use as default gateways. The router advertisements also contain information about the subnet address prefix. The solicitation message is used to ask routers to send advertisements during the advertisement interval. An interesting feature is that a security association between the router and the client can be used to authenticate the client in this stage, when it is entering the network. Also, the used MTU may be distributed using the advertisements.

- Neighbor solicitation and advertisement messages. These are used to connect the link address to IP addresses. A host wishing to find out another host's link address can multicast a neighbor solicitation message (including its own link address), and the queried host should respond with a Neighbor advertisement message containing its link address.
- The router may send a Redirection message to a given host, if it has to handle a packet from the host to a destination, that is better handled by some other gateway in the network (or possibly sent directly)

4.6 IP mobility support

4.6.1 Introduction

The TCP/IP protocol associates every network connection with a unique network address and routes datagrams according to the network part of the address. If a host moves from one network to another, it has to change its network address in order to get its datagrams correctly routed. If it doesn't, the host will be able to send datagrams but not receive any, because the replies are routed to the network associated with the host's address, not to the host's current location.

Additional problems are present when a host changes its network address. All the other hosts that wish to communicate with it have to know the host's network address. The migrating host could try to do a couple of things to inform other hosts of its location such as notifying the routers or updating its DNS entries. Neither of these are adequate, the routers are too many and have a finite amount of memory and the updating of DNS entries may be slow because intermediate name servers cache the previous queries for some time.

The solution to the problems mentioned above is a family of RFCs enabling mobility, commonly called Mobile IP. In IPv4 the Mobile IP protocol is defined more or less as an afterthought on top of the IP protocol which has no support for mobility whatsoever. Because of this Mobile IP in its original version heavily relies on application-level servers to manage the mobility.

In IPv6 support for mobility is defined right into the IP protocol itself, and we will see that the only thing a mobile client needs to be fully mobile is a proxy in the home network, that routes incoming packets to the current location of the host. However, in order to understand the protocol fully, we need to inspect the autoconfiguration features of IPv6.

4.6.2 Autoconfiguration

One of the more elaborate mechanisms built into IPv6 is the autoconfiguration. We want to achieve a mechanism for clients to pop in and out of subnetworks at will, and still have a network address to use. The basic idea is that we define an address space (within the IP network address space) that is based on the MAC address of the network card that we are using. The MAC address is supposed to be unique for the physical device it defines, and thus it can be used as an identifier within the IP address space (more about the details in the addressing section of this paper). This means, that when a client enters an unknown network, it can always present itself using a so called 'link local address', of the form FE80:0:0:0:X:X:X:X, where the X:s denote the part of the address that is filled with the MAC address of the card (slightly expanded into a EUI-64 - number). These addresses are only usable within a single subnet (the routers will discard them), so another approach is to pick the 64 bits at random - the probability of an address conflict within the visited subnetwork is infinitesimally small.

So, having the link local address enables any client to communicate within the subnetwork it enters, but in order to go further we need a protocol called 'stateless autoconfiguration'. In this scheme, the client listens in to the multicast address FF02:0:0:0:0:0:0:1 (all nodes multicast group) while sending a router solicitation on the 'all routers multicast group', FF02:0:0:0:0:0:2 in case there are no frequent router advertisements on the 'all nodes multicast'. The solicitation message will contain the link local address of the client, and all routers in the network are supposed to reply to such a message with a router advertisement. The advertisement message will contain many important bits and pieces of information, but in this case essential is the information whether the client is permitted to perform stateless autoconfiguration at all, and a network prefix, defining the most significant bits of the IP addresses in the current subnetwork. If the clients were permitted to perform autoconfiguration, the client can 'concatenate' the network part of the address and its own MAC address, and use the resulting address as its temporary address - all incoming packets will terminate at the host as the network part of the address will guide them to the right subnetwork. Router advertisements and solicitations are ICMP messages.

4.6.3 Support for mobility

Using autoconfiguration a client can enter any subnet and in principle form a valid IP address that will carry any traffic to and from the client. Unfortunately, the address changes when moving between subnets, and because of this a small addition is needed if we want to keep the client address fixed in the eyes of the corresponding hosts it is communicating with. This feature is achieved using a subheader, the Binding Update option, and a proxy host (home agent) residing in the client's home network. Every time the client moves to a new subnetwork it will send a packet to the home agent (with or without payload) containing the abovementioned binding update header. The header contains the actual (home) address of the roaming client (optional), and the address that the client currently uses (care-of-address). The binding update is an acknowledged message. Because of this location update procedure, the home agent always knows where the mobile client is at the moment (the binding update also contains a lifetime field to protect against old knowledge). The home agent picks up all IP packets that emerge in the home network destined to the mobile client, and forwards them to it using IPv6 tunneling. See fig.15.

Using the same binding update option, the mobile host may also inform any corresponding nodes of a movement to a new subnetwork. Of course, the corresponding nodes may always send their packets to the home network of the mobile client.

Messages originating from the mobile host are sent directly to the corresponding hosts. In this case, the mobile IP sends a 'Home Address' option (containing the mobile node's home address) along with the packet. At the receiving end, the source address will be replaced by the address given in the option's parameters before given to higher network layers, meaning that all applications see the home network address as the peer address.

The messaging supporting mobility is critical from a security point of view, and all messages containing e.g. binding updates should always be properly authenticated using the security features of IPv6.

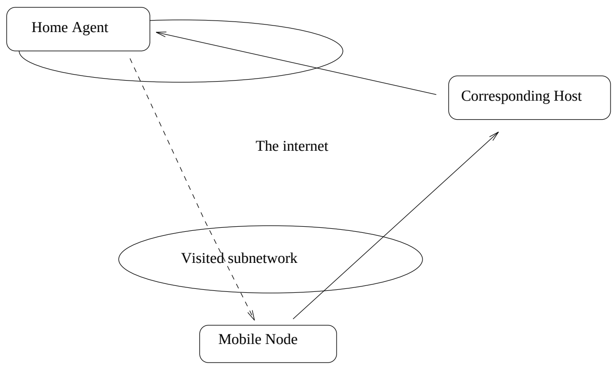


Figure 15: Mobile IP messaging flow (dashed arrow = tunneled connection)

4.7 IPsec

4.7.1 Introduction

During the last few years there has been a clear consensus that data security is a necessary feature for upcoming generations of the internet protocol, and a quite complex security framework has been constructed to protect the packets flowing in the internet. These specifications have been backported to IPv4, which has given developers a chance to test the framework in existing networks during the last year. The specification is, however, still in a state of flux, although the basic architecture and protocols were more or less agreed upon during 1997, and current updates have been minor.

One might argue about placing encryption and authentication (which are the basic security services) in the IP stack itself, which means upgrading the system software - programmers are, wisely enough, reluctant to use any features that are not supported on all software platforms, as this means coding lots of emulation software when porting between operating systems. The driving motivation is that putting these features directly in the IP stack makes them usable for all network applications without any need for special 'encryption support' , and as the new version of the IP requires rewrites of the stacks we anyway come to a situation where all platforms will support security. The chosen algorithms (MD5, DES, 3DES, ...) are more or less tried and true symmetric key encryption and digest algorithms. They are not necessarily watertight against a skilled attacker with huge amounts of computing power (the governments), but the algorithms will not cause a huge computing overhead for the protected packets, and the security they provide should anyway be 'good enough' for all practical purposes.

4.7.2 The protocol

In a nutshell, IPSEC [18] provides access control, authentication, confidentiality, integrity and protection against replays for individual IP datagrams. This is accomplished by using two traffic protecting security protocols, namely the authentication header (AH) and the encapsulating security payload (ESP) in addition to key management procedures and protocols. For a basic structure, see fig.16.

The security association (SA) is a fundamental concept of IPSEC. The security association is a secure logical connection in one direction between two network entities. All IP traffic using the same security association is offered equal protection. A security association is uniquely

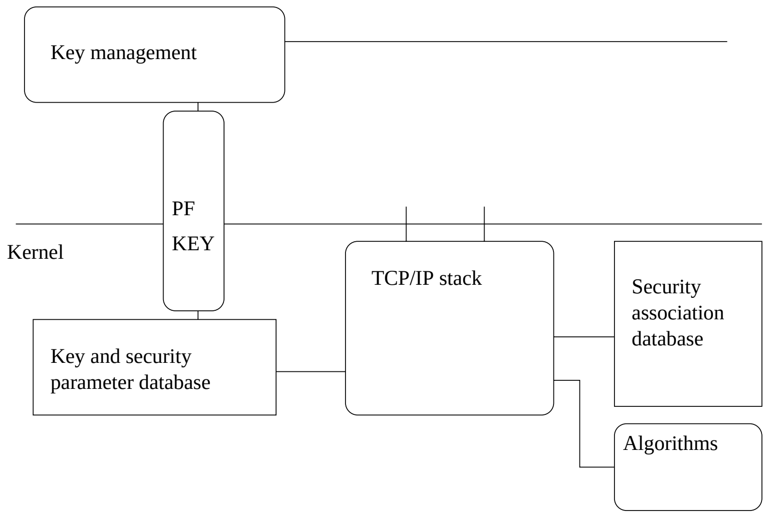


Figure 16: The structure of the IPSEC protocol suite

defined by a triplet, which includes a 32 bit long identifier field called the security parameter index (SPI), a destination IP address and a security protocol which is either AH or ESP. The destination address may be a unicast, multicast or broadcast address but in practice the destination address is a unicast address because the IPSEC key management scheme is not compatible with other address types.

A security association can work in either transport or tunneling mode. In transport mode the security protocol headers are inserted between the IP header and the IP payload. The transport mode is used to protect traffic between two hosts. The tunneling mode is similar to IPv6 tunneling, as the complete IP datagram with header and payload is encapsulated (encrypted/authenticated) by the security protocol as the payload of another IP datagram. The tunneling mode makes it possible to create a protected network from two subnets separated by an unsecure network by adding a security gateway with IPSEC capabilities in both subnets. Configured as the default routers for the respective subnets the gateways route traffic directed to the other subnet protecting the communications with tunnel mode security associations. A security gateway has to support only tunneling mode security associations whereas a host has to support both modes.

Whenever more than one security protocol (any combination of ESPs and AHs) is needed to satisfy the current security policy there will be several security associations that have to be set up for the IP packets. These individual security associations in the security association bundle are completely independent of each other and may therefore end at different network locations. A security association bundle is created by combining transport adjacency with iterated tunneling. Transport adjacency refers to the process of applying more than one security protocol in transport mode on the IP datagrams. As this implies that the end points of the security associations remain the same, the only appropriate setup for transport adjacency bundling is to apply AH authentication after ESP encryption.

Iterated tunneling refers to applying one or more tunnel mode security associations on an IP datagram. This enables any number of security associations to originate and terminate at different points on the network path connecting the ultimate endpoints. With iterated tunneling there are no strict ordering requirements of the security protocols as with transport adjacency.

Creation and use of security associations is controlled by two nominally separate databases, the security policy database (SPD) and the security association database (SAD). The former defines the security association needed for incoming and outgoing IP traffic and the latter contains the specific parameters for each security association. Both databases have to discriminate between different network interfaces (IP address, port) as well as between incoming and outgoing traffic.

The security policy database contains the criteria for the protection offered to different datagrams. Each datagram has one of three processing choices: it may be protected by IPSEC, it may pass without further protection or it may be discarded. The selection rules may contain wildcards that match any value, and in the case of an IP address the rule may be specified as a range or as a network address and netmask pair. The entries in the security policy database have to have an internal order so that fine and coarse grained security policy entries overlapping each other may be present at the same time.

In the security policy database each entry requiring IPSEC protection contains a pointer to list of active security associations in the security association database. These entries, discriminated by the destination IP address, the IPSEC protocol, and the security parameter index, also contain the sequence number counter, the sequence number counter overflow flag, the anti-replay counter for inbound datagrams, the AH authentication algorithms and its parameters, the ESP encryption and authentication algorithm and its parameters, the hard and soft lifetimes, the protocol mode and path MTU value along with path MTU aging information. The sequence number counter and the 32 bit anti-replay counter are used to ensure that the datagrams are not received more than a predefined amount out of order thus prohibiting replays of old datagrams. The soft lifetime is used to inform that the security association's lifetime is about to end and that a new security association is to be negotiated. The hard lifetime indicates the final expiration of the security association. Both lifetimes may be measured in either seconds or number of bytes transmitted.

An outbound IP datagram is first matched against the entries in the security policy database. If the entry requires IPSEC processing, a lookup is made at the relevant position in the security association database. If no matching security association is found, a new security association has to be inserted into the database by some key exchange protocol. This functionality can be implemented using a protocol family named PF_KEY, that enables a daemon to listen for security association requests from the IPSEC stack, and to insert these new security associations at run-time. When the security association has been inserted, the appropriate security protocols are applied to the datagram using the algorithms and parameters defined within the security association.

An inbound IP datagram is matched against the security association database using the previously mentioned triplet as the table lookup key. The IPSEC processing is applied to the IP datagram using the security association's parameters until all the security associations are processed. When the IPSEC processing is complete the IP datagram and the security association processing order is looked up in the security policy database to ensure that the corresponding security policy exists for the IP datagram. If a match is not found the packet is discarded, otherwise the processing continues, e.g. a tunneled IP datagram is forwarded to its destination.

4.7.3 Authentication header

The authentication header [19] security protocol provides authentication and integrity as well as replay protection for individual IP datagrams by inserting the authentication header between the original IP header and the payload. Both transport and tunnel mode security associations are supported. No encryption is applied to the datagram, the payload is sent in plaintext. The authentication header protocol performs the integrity check calculation over the authentication header and IP datagram's payload and header fields except for the type of service, flags, fragment offset, time to live and header checksum fields because they may be changed in transit. The authentication header includes a four octets long sequence number that provides protection against replays of old datagrams. The currently defined message authentication mechanisms rely on the keyed hashing message authentication (HMAC) framework and use MD5, RIPEMD or SHA as the authentication algorithms.

4.7.4 Encapsulating security payload

The encapsulating security payload provides the security services authentication, confidentiality, integrity and protection against replays for IP datagrams. In this payload type encryption and integrity checking can be applied independently of each other, e.g. the payload may be encrypted without any any integrity checking. Both transport and tunnel mode security associations are supported. The replay protection is done in the same fashion as in the authentication header protocol, each encapsulating security payload contains a sequence number to protect against replays. The payload data is transported inside the encapsulating security payload protocol and padded into proper length for the encryption algorithm. The integrity check calculation is done only on the encapsulating security payload and is applied after encryption. The authentication and integrity calculation uses the same framework and algorithms as the authentication header protocol. The currently defined encryption algorithms are DES with an explicit initialization vector, DESX, triple DES, RC5, IDEA, triple IDEA, CAST, BLOWFISH and RC4 all of which work in cipher block chaining (CBC) mode.

4.7.5 Key management

Several key management protocols have been proposed for the purpose of finding keys for a given security association on demand. Three main contenders have somewhat different approaches to the problem at hand. The Photuris protocol is more or less Diffie-Hellman key exchange between the parties wishing to communicate. This means that the keys are generated peer-to-peer and on-the-fly, without any intermediate security servers or other entities. Unfortunately, as the mechanism doesn't use any key directory or third-party authentication it is subject to man-in-the-middle attacks, where a third party can, by participating in the initial key generation, pretend to be the other party to either of the peers.

The second contender is Sun Microsystems's Simple Key-management for Internet Protocols (SKIP) protocol series. This is based on Diffie-Hellman as well, but is not susceptible to man-in-the-middle attacks, as the 'seeds' for the keys are published in a trusted, third-party directory. The main drawback of this protocol is that the keys generated for communication between two peers will be static (non- changing), but that could be remedied by switching to another session key, negotiated in the initial, secure channel, as soon as possible.

The third, and the strongest candidate, is the ISAKMP/Oakley - protocol (Internet Security Association and Key Management protocol), which is a very generic (and extensible) framework for establishing session keys. Although Diffie-Hellman is an option in this protocol as well, several other modes support e.g. mutual authentication by means of X.500-public keys, or the use of pre-shared keys for the initiation of the protocol. Of the two standard proposals, ISAKMP defines a message set for performing all kinds of security - related functionality, and Oakley is a proposal for session key setup (mutual authentication) within this framework.

4.8 IPv6 transition

IPv6 based systems must coexist with the installed IPv4 systems for an extended period. In this dual internetworking protocol environment IPv4 and IPv6 routing infrastructure will be mixed. In the beginning the IPv6 environments will probably be isolated islands and must communicate over IPv4-only routing regions. In the end of the transition the picture will be the opposite.

In the basic dual-IP scheme routers may independently support IPv4 and IPv6 routing. Forwarding of IPv4 packets is based on routes learned through IPv4 specific routing protocols. Similarly, IPv6 packet routes are learned through IPv6 specific protocols. This means that separate instances of the routing protocols are used for the IP versions. A integrated routing protocol to support both versions of IP is not yet available.

Tunneling techniques must be extensively used in this mixed environment. Dual hosts and routers can tunnel IPv6 packets over regions of IPv4 routing by encapsulating them within IPv4 packets. For the tunneled protocol a tunnel appears as a single hop point to point link. Two flavours of tunneling are to be used. When the tunnel endpoint address is determined from the configuration of the encapsuling node, one speaks of *configured tunneling.* The tunnel endpoint address must be stored for each tunnel and this address will be used as the destoination address for the encapsulating IPv4 header.

The use of a configured 'default tunnel' may be used by nodes connected to a IPv4 infrastructure to reach the IPv6 'backbone'. In this case the destination address of the encapsulating IPv4 header is the IPv4 address of a IPv6/IPv4 router bordering the backbone. This tunnel can be configured into the routing table as the default route. An other possibility for the tunnel endpoint is to be a IPv4 'anycast address'. With this approach several IPv6/IPv4 routers advertise to the same IPv4 address. All of these routers accept packets to this addres as their own ad will decapsulate them. When a IPv6/IPv4 node sends an encapsulated packet to the address it will only be delivered to the one of the border routers. The IPv4 routing is assumed to take care of carrying the traffic to the closest router. Tunneling to an 'anycast address' offers a high degree of robustness since the normal fallback mechanisms of IPv4 routing can be used.

In *automatic tuneling* the endpoint is determined from the packet being tunneled. For automatic tunneling to be possible the destination has to be a IPv4 comaptible IPv6 address. The 32 low order bits are extracted and used a tunnel endpoint address.

4.9 DNS extensions

The current system for storage of Internet addresses has to be adapted to support IPv6 as applications assume that address queries return 32-bit IPv4 addresses only[2] . Clearly a new RR type is needed to map a domain name to an IPv6 address. Also a new domain has to be

[2]This entire section is based on work in progress

defined to support lookups based on the address. Finally the existing queries that perform additional section processing to locate IPv4 addresses i.e. NS, MX, and MB have to made IPv6 aware.

The new RR type, AAAA stores the least significant bits of a single IPv6 address. It also contains the domain name of another IPv6 system, typically one that describes a complete link or site. The most significant bits will be copied from the address of that system. If is has many addresses the lsbs will be combined with each prefix of the several addresses resulting in as many IPv6 addresses. If a system is connected to several domains, e.g. a system connected to several providers, it may need several records.

A new domain `ip6.int` is defined to map the IPv6 addresses to host names. An address is represented by a sequence of nibbles separated by dots with the suffix `ip6.int`, each nibble represented by a hexadecimal digit. The nibbles are in reversed order. Thus an address `4321:0:1:2:3:4:567:89ab` would correspond to

```
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.
```

It should be noted that the network renumbering is easy and only requires updating a single DNS record. however, it must be synchronized with address configuration and router renumbering procedures. At the moment no procedures for this are ready.

References

[1] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, IETF, 1987.

[2] P. Mockapetris. Domain names - implementation and specification. RFC 1035, IETF, 1987.

[3] S. Feit. *TCP/IP Achitecture,protocols and implementation with IPv6 and IP security*, chapter 5,12. McGraw-Hill, 2nd edition, 1996.

[4] Määräys Internet-tietoverkon suomalaisista verkkotunnuksista, May 1997.

[5] Määräys suomalaisen verkkotunnuksen määrittelystä Internet-tietoverkon nimipalvelimiin, May 1997.

[6] N. McKay. Network solutions hangs on. *Wired News*, October 1998.

[7] J. Zittrain. Keyword:obsolete. *Wired*, page 93, September 1998.

[8] C. Oakes. Internet keywords patent spat. *Wired News*, July 1998.

[9] D. Eastlake and C. Kaufman. Domain name system security extensions. RFC 2065, IETF, 1997.

[10] P. Vixie, S. Thomson, Y. Rekhter, and J.Bound. Dynamic updates in the Domain Name System. RFC 2136, IETF, 1997.

[11] D. Eastlake. Secure domain name system dynamic update. RFC 2137, IETF, 1997.

[12] http://www.umich.edu/ dirsvcs/ldap/doc/, Sep 98.

[13] http://www.netscape.com, Sep 98.

[14] J. Myers. Simple authentication and security layer (sasl). RFC 2222, Netscape, 1997.

[15] M. Wahl. A summary of the x.500(96) user schema for use with ldapv3. RFC 2256, Critical Angle Inc., 1997.

[16] M. Miller. *Ipmplementing IPv6*. M&T Books, 1st edition, 1998.

[17] C. Huitema. *IPv6, the new Internet Protocol*. Prentice Hall, 2nd edition, 1998.

[18] R. Atkinson. Security architecture for the internet protocol. RFC 1825, Naval Research Laboratory, 1995.

[19] R. Atkinson. Ip authentication header. RFC 1826, Naval Research Laboratory, 1995.