To support the storage of IPv6 addresses the following extensions are defined:

- A new resource record type is defined to map a domain name to an IPv6 address.
- A new domain is defined to support lookups based on address.
- Existing queries that perform additional section processing to locate IPv4 addresses are redefined to perform additional section processing on both IPv4 and IPv6 addresses.

### AAAA Record Type

The AAAA resource record type is a new record specific to the Internet class that stores a single IPv6 address.

A 128 bit IPv6 address is encoded in the data portion of an AAAA resource record in network byte order (high-order byte first).

An AAAA query for a specified domain name in the Internet class returns all associated AAAA resource records in the answer section of a response.

A type AAAA query does not perform additional section processing.

The textual representation of the data portion of the AAAA resource record used in a master database file is the textual representation of a IPv6 address.

### IP6.INT Domain

A special domain is defined to look up a record given an address.

The domain is rooted at IP6.INT.

An IPv6 address is represented as a name in the IP6.INT domain by a sequence of nibbles separated by dots with the suffix "IP6.INT". The sequence of nibbles is encoded in reverse order, i.e. the low-order nibble is encoded first, followed by the next low-order nibble and so on. Each nibble is represented by a hexadecimal digit.

Example:

The inverse lookup domain name corresponding to the address 4321:0:1:2:3:4:567:89ab would be b.a.9.8.7.6.5.0.4.0.0.0.0.0.0.0.0.0.1.2.3.4.IP6.INT.

### A6 Resource Record

The A6 RR contains two or three fields:

- A prefix length.
- An IPv6 address suffix.
- The name of the prefix.

The domain name component shall not be present if the prefix length is zero.

The address suffix component shall not be present if the prefix length is 128.

It is suggested that an A6 record intended for use as a prefix for other A6 records have all the insignificant trailing bits in its address suffix field set to zero.

Example (of textual representation):

$ORIGIN example2.net.
subnet5         A6      48 0:0:0:1:: ipv6net2.example2.net.
ipv6net2        A6      0  6666:5555:4::

### Binary Labels

A "Bit-String Label" may appear within domain names.

Represents a sequence of "One-Bit Labels".

Enables RRs to be stored at any bit-boundary in a binary-named section of the domain name tree.

Are intended to efficiently solve the problem of storing data and delegating authority on arbitrary boundaries (for reverse zones).

Textual Representation example:

```
[1011010000011101] [64072/14] [1d/64] [35/208.0.0.0/8]
```
IP v6 DNS Extensions

IP v6 DNS Extensions - IP v6 ARPA Domain

- A new special domain is defined to look up a record given an address.
- The domain is rooted at IP6.ARPA.
- This new scheme for reverse lookups relies on Binary Labels.
- The inverse lookup domain name corresponding to the address 4321:0:1:2:3:4:567:89ab would be \[x432100000001000200030004056789ab].IP6.ARPA.
- DNS address space delegation is implemented not by zone cuts and NS records, but by the new DNAME resource record.

IP v6 DNS Extensions - Non-Terminal DNS Name Redirection

- A new RR called “DNAME” provides the capability to map an entire subtree of the DNS name space to another domain.
- It’s a solution to the problem of maintaining address-to-name mappings in a context of network renumbering.
- Renumbering Example:
  - From: 20aa:00bb:cccc:dddd:1234:5678:1212:5675
  - $ORIGIN [x20aa00bbcccc/48].ip6.arpa.
  - [xddd/16] DNAME ipv6-rev.example.com.
  - [xddd/16] DNAME ipv6-rev.example.com.
  - $ORIGIN ipv6-rev.example.com.
  - [x1234567812125675/64] PTR host.example.com.

IP v6 DNS Extensions - Modifications to existing Query Types

- All existing query types that perform type A additional section processing, must be redefined to perform type A, A6 and AAAA additional section processing, i.e.:
  - Name server (NS)
  - Mail exchange (MX)
  - Mailbox (MB)
- These new definitions mean that a name server must add any relevant IPv4 addresses and any relevant IPv6 addresses available locally to the additional section of a response when processing any one of the above queries.

Transition Mechanisms - Dual IP Stacks

- Is the simplest mechanism for IPv4 and IPv6 coexistence.
- Node has both IPv4 and IPv6 stacks and addresses.
- DNS Resolver returns IPv6, IPv4 or both to application.
- IPv6 applications can communicate with IPv4 nodes.

Transition Mechanisms - Tunneling IPv6 in IPv4

- IPv6 encapsulated in IPv4
- Four possible configurations:
  - Router-to-Router
  - Host-to-Router
  - Host-to-Host
  - Router-to-Host
- The tunnel endpoints take care of the encapsulation. This process is “transparent” to the other nodes.
- The manner in which endpoints addresses are determined defines:
  - Configured tunnels
  - Automatic tunnels
  - Multicast tunnels
Transition Mechanisms

**Configured Tunneling**

- Tunnel endpoints are fixed (manually configured).
- Tunnel endpoints must be dual-stack nodes.
  - The IPv4 address is the endpoint for the tunnel.
  - Require reachable IPv4 addresses.
- The tunnels can be either unidirectional or bidirectional.
  - Bidirectional configured tunnels behave as virtual point-to-point links.

**Automatic Tunneling**

- IPv4 tunnel endpoint address is determined from the IPv4-compatible destination IPv6 address.
  - Example: ::170.210.79.4
  - Terminates on a host.
  - Routing table redirects ::/96 to automatic tunneling interface.
  - If two hosts have IPv4-compatible IPv6 addresses, they can communicate across an IPv4 infrastructure using automatic tunneling.
  - A dual router, upon receiving an IPv6 packet destined for a host with an IPv4-compatible address, can automatically tunnel that packet to its endpoint.
**Transition Mechanisms**

**Multicast Tunneling: 6over4**

- Interconnection of isolated IPv6 domains in an IPv4 world.
- No explicit tunnels.
- The egress router must:
  - Have a dual stack
  - Have a globally routable IPv4 address
  - Have an IPv4 multicast infrastructure
  - Implement 6over4 on an external interface
- Uses IPv4 as a link layer for IPv6, that’s why IPv4 multicast is needed.

**Transition Routing**

- Terms related to transition routing architecture:
  - **Border router**: A router that forwards packets across routing domain boundaries.
  - **Routing domain**: A collection of routers that coordinate routing knowledge using a single protocol.
  - **Routing region**: Collection of routers, interconnected by a single Internet protocol, that coordinate their routing knowledge using routing protocols from a single IP stack. A routing region may be a superset of a routing domain.
  - **Reachability information**: Information describing the set of reachable destinations that can be used for packet forwarding decisions.
  - **Route leaking**: Advertisement of network layer reachability information across routing boundaries.

---

**Transition Routing Routing Example (1)**

**Region A: IPv6/v4 routers**

- **Region B: IPv4-only routers**

**Transition Routing Routing Example (2)**

**Region A: IPv6/v4 routers**

- **Region B: IPv4-only routers**

**Transition Routing Routing Example (3)**

**Region A: IPv6/v4 routers**

- **Region B: IPv4-only routers**

**Transition Routing Routing Example (4)**

**Region A: IPv6/v4 routers**

- **Region B: IPv4-only routers**
**6to4**
- Mechanism for IPv6 sites to communicate with each other over the IPv4 network without explicit tunnel setup.
- Allows communication with native IPv6 domains.
- Assigns an interim unique IPv6 address prefix to any site that currently has at least one globally unique IPv4 address.
- Not requires:
  - IPv4-compatible IPv6 addresses
  - configured tunnels
- Uses the prefix 2002::/16 to form 6to4 prefixes derived from the IPv4 Address.

<table>
<thead>
<tr>
<th>37</th>
<th>32</th>
<th>16</th>
<th>64 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>IPv6</td>
<td>SA ID</td>
<td>Interface ID</td>
</tr>
</tbody>
</table>

**6to4 – Terminology**
- Requires an IPv4 network communicating both 6to4 routers.
- 6to4 prefix: a prefix derived from an IPv4 address.
- 6to4 address: an IPv6 address constructed using a 6to4 prefix.

**6to4 – Scenario: All sites work the same**
- Requires an IPv4 network communicating both 6to4 routers.
- Each site has an IPv4 prefix in the form 2002:WWXX:YYZZ::/48
- Outgoing packets are encapsulated into IPv4 at the 6to4 router.
- Incoming packets are decapsulated and sent to the internal IPv6 network.
- Any number of 6to4 sites can interoperate with no tunnel configuration.

**Transition Routing Summary**

<table>
<thead>
<tr>
<th>Host A</th>
<th>Host B</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>v4-compatible address no local v6 router</td>
<td>v4-compatible address no local v6 router</td>
<td>host to host tunneling in both directions</td>
</tr>
<tr>
<td>v4-compatible address no local v6 router</td>
<td>v4-compatible address local v6 router</td>
<td>A→B: host to host tunnel B→A: v6 forwarding plus router→host tunnel</td>
</tr>
<tr>
<td>v4-compatible address no local v6 router</td>
<td>incompatible address local v6 router</td>
<td>A→B: host to router tunnel plus v6 forwarding B→A: v6 forwarding plus router to host tunnel</td>
</tr>
<tr>
<td>6to4 address</td>
<td>Any IPv6 address</td>
<td>6to4</td>
</tr>
<tr>
<td>v4-compatible address local v6 router</td>
<td>v4-compatible address local v6 router</td>
<td>end to end native v6 in both directions</td>
</tr>
<tr>
<td>v4-compatible address local v6 router</td>
<td>incompatible address local v6 router</td>
<td></td>
</tr>
<tr>
<td>incompatible address local v6 router</td>
<td>incompatible address local v6 router</td>
<td></td>
</tr>
</tbody>
</table>

**SIIT: Stateless IP/ICMP Translation**
- Allows IPv6-only hosts to talk to IPv4 hosts.
- Header translator maps corresponding header fields of IPv4 to IPv6.
- Requires one temporary IPv4 address per host.
- Problem: if no corresponding fields/infos in both headers => no translation possible.
- Conclusion: except segmentation no usage of IPv6 extension headers.
- Requires IPv4-mapped IPv6 address ::FFFF:d.d.d.d
### NAT-PT: Network Address Translation – Protocol Translation

- Enables communication between pure IPv6 and IPv4 nodes.
- Combines two techniques: NAT (Network Address Translation) and SIIT (Protocol Translation).
- Requires at least one IPv4 address per site.

**Operation:**
- IPv6 node sends packet to NAT-PT server with special destination address.
- NAT-PT server manages pool of IPv4 addresses, translates headers: IPv4 to IPv6. Assigns IPv4 address to IPv6 address, forwards packet to IPv4 node.
- IPv4 node: first IPv4 address of IPv6 node has to be received from DNS. DNS server requests NAT-PT to assign and delivers reserved IPv4 address of IPv6 node.

### Traditional NAT-PT

- Traditional NAT-PT would allow hosts within a V6 network to access hosts in the V4 network.
- In a traditional NAT-PT, sessions are unidirectional, outbound from the V6 network.
- This is in contrast with Bi-directional NAT-PT, which permits sessions in both inbound and outbound directions.
- There are two variations to traditional NAT-PT:
  - Basic NAT-PT: a block of V4 addresses are set aside for translating addresses of V6 hosts as they originate sessions to the V4 hosts in external domain.
  - NATPT-PT, which stands for "Network Address Port Translation + Protocol Translation", would allow V6 nodes to communicate with the V4 nodes transparently using a single V4 address.

### Bi-directional NAT-PT

- Sessions can be initiated from hosts in V4 network as well as the V6 network.
- V6 network addresses are bound to V4 addresses:
  - Statically
  - Dynamically
- Hosts in V4 realm access V6 realm hosts by using DNS for address resolution.
- A DNS Application-Level-Gateway must be employed to facilitate name to address mapping.

### Bump in the Stack

- IPv4 applications can transparently communicate via an IPv6 net (if application uses logical names and DNS service).
- Inserts 3 additional modules into IPv4 protocol stack (Dual Stack Host).
- Operation:
  - Translator maps IPv4 packets into IPv6 packets using protocol translation (SIIT).
  - Extension name resolver creates DNS requests (A-rec. + AAAA-rec.) upon appl. DNS request.
  - If DNS server replies A-rec., this is guided directly to IPv4 application.
  - If DNS server replies only AAAA-record, address mapper reserves IPv4 address.
  - Then A-rec. is derived (from reserved IPv4 address and AAAA-rec.) and given to application.
  - Address mapper manages pool of IPv4 addresses and assigned IPv6 addresses.

### IPSec – Network Security

- IPSec is designed to provide interoperable, high quality, cryptographically-based security for IPv4 and IPv6.
- IPSec provides security at the IP layer, transparent to applications.
- IPSec offers services for:
  - Authentication: Authenticate the sender.
  - Confidentiality: Encrypt data before transmission.
  - Data Integrity: Detect altered data in packets.
  - Anti-Replay: Detect replayed packets.
  - Open standard, published by the IETF.

### IPSec – Protocols

- IPSec uses two protocols to provide traffic security.
  - **Authentication Header (AH):**
    - Connectionless integrity.
    - Data origin authentication.
    - Anti-replay service (optional).
  - **Encapsulating Security Payload (ESP):**
    - Connectionless integrity.
    - Data origin authentication.
    - Anti-replay service (optional).
    - Confidentiality (encryption).
    - Limited traffic flow confidentiality
  - Two modes of use: transport or tunnel.
### IPsec – Security Associations

The concept of a “Security Association” is fundamental to IPsec.

- A Security Association (SA) is a simplex “connection” that affords security services to the traffic carried by it.
- A Security Association is unidirectional.
- A Security Association is identified by a triple consisting of:
  - Security Parameter Index (SPI).
  - IP Destination Address.
  - Security protocol identifier (AH or ESP).

### IPsec – Security Databases

There are two nominal databases in this model:

- Security Policy Database
  - Specifies the policies that determine the disposition of all IP traffic inbound or outbound from an IPsec implementation.
  - An SPD must discriminate among traffic that is afforded IPsec protection and traffic that is allowed to bypass IPsec.
- Security Association Database
  - Contains parameters that are associated with each security association.
  - Selector: a set of IP and upper layer protocol field values that is used by the SPD to map traffic to a policy, i.e., an SA.

### IPsec – Basic Overview

#### Case 1: SA already available.

IPSec packet processing:

- Look up in the Security Policy Database (SPD) how to handle the packet:
  - Discard
  - Bypass IPsec -> use IP
  - Apply IPsec

Secure transmission possible?

- Lookup in the Security Association List (SA List) if a Security Association (SA) is available, i.e., if a secure transmission is possible.
  - SA stores information about Authentication and / or Encryption algorithm and symmetric, shared keys.
IPSec – Basic Overview
Case 2: SA not available.

IPSec packet processing:
- Dynamically create a SA using Internet Key Exchange (IKE)
- Exchange shared keys for IP Sec

Secure transmission possible?

Host
TCP
MAC
IPSec

Internet

SPD

Host
TCP
MAC
IPSec

Internet

IPSec packet processing:
- Internet Key Exchange (IKE)
- Create a IKE SA using public keys
- Exchange shared keys for IP Sec

Secure key exchange possible?

Host
TCP
MAC
IPSec

Internet

SPD

IKE SA

Host
TCP
MAC
IPSec

Internet

Negotiate algorithms and keys for IP Sec

Host
TCP
MAC
IPSec

Internet

SA

Host
TCP
MAC
IPSec

Internet

SA

Host
TCP
MAC
IPSec
IPsec – Supported Combinations

Host to Host

- IP_AH_payload (transport)
- IP_ESP_payload (transport)
- IP_AH_ESP_payload (transport)
- IP (host)_AH_IP_payload (tunnel)
- IP (host)_ESP_IP_payload (tunnel)

Internet

Security Gateway to Security Gateway

- IP (SG)_AH_IP_payload (tunnel)
- IP (SG)_ESP_IP_payload (tunnel)

Internet

Combination of Cases

- Two options (tunnel)
- Three options (transport)
- Two options (tunnel)

Remote Access

- Two options (tunnel)
- Three options (transport)
- Two options (tunnel)

IPsec – Authentication header

- Authentication of data origin
- Data integrity
- Anti-replay (optional)

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Header</td>
<td>Payload length</td>
<td>Reserved</td>
</tr>
<tr>
<td>Security Parameters Index (SPI)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number Field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authentication Data (variable)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPI = 0 is forbidden, 1..255 is reserved
Seq. Number only increases (no reset to 0) for anti-replay

IPsec – Encapsulation Security Payload

- Data integrity
- Data encryption
- Authentication (optional)
- Anti-replay (optional)

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Parameters Index (SPI)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number Field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload Data (variable)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Padding (0..255 bytes)</td>
<td>Padding length</td>
<td>Next Header</td>
</tr>
<tr>
<td>Authentication Data (variable)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPI = 0 is forbidden, 1..255 is reserved
Seq. Number only increases (no reset to 0) for anti-replay
IP v6
IPSec – AH Transport Mode
Before applying AH
IP Header Optional headers TCP, UDP, ICMP, etc. Data

After applying AH
Original IP Header Optional headers(*) AH Optional headers(**) TCP, UDP, ICMP, etc. Data

Authenticated except for mutable fields

(**): Dest. Opt

IP v6
IPSec – AH Tunnel Mode
Before applying AH
IP Header Optional headers TCP, UDP, ICMP, etc. Data

After applying AH
New IP Header New optional headers AH Original IP Header Optional headers TCP, UDP, ICMP, etc. Data

Authenticated except for mutable fields in new IP hdr

IP v6
IPSec – ESP Transport Mode
Before applying ESP
IP Header Optional headers TCP, UDP, ICMP, etc. Data

After applying ESP
Original IP Header Optional headers(*) ESP Optional headers(**) TCP, UDP, ICMP, etc. Data ESP Trailer ESP Auth

Encrypted
Authenticated

(**): Dest. Opt

IP v6
IPSec – ESP Tunnel Mode
Before applying ESP
IP Header Optional headers TCP, UDP, ICMP, etc. Data

After applying ESP
New IP Header New optional headers ESP Original IP Header Optional headers TCP, UDP, ICMP, etc. Data ESP Trailer ESP Auth

Encrypted
Authenticated

IP v6
IPSec – AH-ESP Transport Mode
Before applying AH-ESP
IP Header Optional headers TCP, UDP, ICMP, etc. Data

After applying AH-ESP
Original IP Header Optional headers(*) AH Optional headers(**) TCP, UDP, ICMP, etc. Data ESP Trailer ESP Auth

Authenticated except for mutable fields

(**): Dest. Opt

IP v6
IPSec – Example

PC_1 PC_2

* * *

AH

TCP, UDP, ICMP, etc.

Data

Original IP Header Optional headers(*) AH Optional headers(**) TCP, UDP, ICMP, etc. Data ESP Trailer ESP Auth

Encrypted
Authenticated

IP v6
IPSec – Example


Insert one table here


Insert another table here

PC_1 PC_2

* * *

AH

TCP, UDP, ICMP, etc.

Data

Original IP Header Optional headers(*) AH Optional headers(**) TCP, UDP, ICMP, etc. Data ESP Trailer ESP Auth

Encrypted
Authenticated

IP v6
IPSec – Example


Insert a third table here
### Mobility Terms

- **Home address:** permanent address of the mobile node.
- **Home subnet prefix:** prefix corresponding to the home address.
- **Foreign subnet prefix:** prefix of the foreign link.
- **Care-of address:** address assigned to the mobile node on the foreign link.
- **Binding:** the association of the home address of a mobile node with a care-off address.

### Mobility – Home Binding Procedure

- The mobile node obtains its care-of address.
- The mobile node sends a Binding Update message to the home agent.
- The home agent replies by returning a Binding Acknowledgement message.
- The home agent intercepts packets destined to the mobile node and tunnels them to the care-of address. The mobile node reverse tunnels traffic destined to the correspondent node.

### Mobility – Communication without binding

- **Sending Packets**
  - The packet is sent to the home agent using IPv6 encapsulation.
  - The home agent decapsulates the tunneled packet and forwards it towards the correspondent node.
- **Receiving Packets**
  - The correspondent node sends the packet to the home network.
  - The home agent intercepts the packet.
  - The home agent encapsulates the packet using IPv6 encapsulation and sends it to the mobile node care-of address.

### Mobility – New IPv6 Protocol: Mobility Header

- Four messages to perform the Return Routability Procedure
  - Home Test Init (HoTI)
  - Home Test (HoT)
  - Care-of Test Init (CoTI)
  - Care-of Test (CoT)
- Four messages to manage the bindings
  - Binding Update
  - Binding Acknowledgement
  - Binding Refresh Request
  - Binding Error

### Mobility – New ICMP Messages

- Messages use in the dynamic home agent address discovery mechanism.
  - Home Agent Address Discovery Request
  - Home Agent Address Discovery Reply
- Messages used for network renumbering and address configuration on the mobile node
  - Mobile Prefix Solicitation
  - Mobile Prefix Advertisement

### Mobility – New IPv6 Destination Option

- Mobile IPv6 defines a new IPv6 destination Option, the **Home Address** destination option.
- This option is used in a packet sent by a mobile node while away from home, to inform the recipient of the mobile node’s home address.
Mobile IPv6 defines a Routing Header to carry the Home Address for packets sent from a correspondent node to a mobile node.

This Routing Header type (Type 2) is restricted to carry only one IPv6 address.

**Binding Cache**
- Maintained by each IPv6 node.
- A separate Binding Cache maintained for each of the node’s IPv6 addresses.
- When sending a packet, the BC is searched before the Destination Cache.
- Entries marked as “home registration” or “correspondent registration”

**Binding Update List**
- Maintained by each mobile node.
- Records information for each BU sent by the node.
- Includes bindings sent to:
  - Correspondent Nodes
  - Home Agent
  - Home Agent on a previous foreign link

**Home Agents List**
- Maintained by each home agent and each mobile node.
- Records information about each home agent from which this node has received a Router Advertisement in which the Home Agent bit is set.
- This list is similar to the Default Router List (Neighbor Discovery).
- Used by a Home Agent in the dynamic home agent discovery mechanism.
- Enables a node to notify a home agent on its previous foreign link.

**Node Keys**
- Each correspondent node has a secret key, Kcn.
- The node uses this key to verify the cookies.
- This key does not need to be shared with any other entity.
- A correspondent node generates Kcn each time it boots.
- Kcn consists of 20 octets.

**Nonces**
- Each correspondent generates nonces at regular intervals.
- Generated using a random number generator.
- Each nonce is identified by a nonce index.
- A correspondent node may use the same Kcn and nonce with all the mobiles it is in communication with.
- Nonce is an octet string of any length. Recommended length is 64-bit.
Mobility – Cookies

- Cookies sent to the correspondent node.
  - Generated Randomly.
  - Used to verify that the response matches the request.
  - HoT and CoT cookies
- Cookies sent to the mobile node. Produced cryptographically from nonces.
  - Home Cookie
  - Care-of Cookie

Mobility – Cryptographic Functions

- MAC_K(m)
  - Computed on message m with key K.
  - HMAC SHA1.

- Hash(m)
  - Hash of message m.
  - SHA1.

Mobility – Return Routability Procedure

Session Key
Kbu = hash(home cookie | care-of cookie)

Mobility – Binding to a Correspondent Node

Binding Update
Src= care-of address
Dst= correspondent
Parameters:
- home address
- MAC_Kbu(care-of address | correspondent node address | BU)
- home nonce index
- care-of nonce index
- sequence number
- ...(more fields, not security related)

Binding Acknowledgement
Src= correspondent
Dst= care-of address
Parameters:
- sequence number
- MAC_Kbu(care-of address | correspondent node address | BA)
- ...

Mobility – Route Optimization

- Sending Packets
  - The packet is sent to the correspondent using the Home Address Destination Option. Source Address: care-of address.
  - The correspondent node swaps the IPv6 Source address and the Home Address Destination Option.
- Receiving Packets
  - The correspondent node sends the packet using the Routing header. IPv6: Dest. Address = care-of address included in the routing header.
  - The mobile node receives the packet, swaps the Dest. Address and the IPv6 header. Resubmits the packet for IP processing.

Socket Interface Extensions for IPv6

Motivation

- While IPv4 addresses are 32 bits long, IPv6 interfaces are identified by 128-bit addresses.

  - The socket interface makes the size of an IP address quite visible to an application.

  - Those parts of the API that expose the addresses must be changed.

  - IPv6 also introduces new features which must be made visible to applications via the API, e.g.:
    - Traffic class
    - Flow Label
Socket Interface Extensions for IPv6

Design Considerations

- The API changes should:
  - Provide both source and binary compatibility for programs written to the original API.
  - Be as small as possible in order to simplify the task of converting existing IPv4 applications to IPv6.
  - Be able to use this API to interoperate with both IPv6 and IPv4 hosts. Applications should not need to know which type of host they are communicating with.

What Needs to be Changed

- Core socket functions
  - These functions need not change for IPv6.
- Address data structures
  - A new IPv6-specific address data structure is needed.
- Name-to-address translation functions
  - New functions are defined to support IPv4 and IPv6.
  - The POSIX 1003.1g draft specifies a new nodename-to-address translation function which is protocol independent.
- Address conversion functions
  - New functions that convert both IPv4 and IPv6 addresses.
- Miscellaneous features
  - New interfaces to support the IPv6 traffic class, flow label, and hop limit header fields.
  - New socket options are needed to control the sending and receiving of IPv6 multicast packets.

IPv6 Address Family and Protocol Family

- New address family name: AF_INET6
  - Defined in <sys/socket.h>
  - New sockaddr_in6 data structure.
- New protocol family name: PF_INET6
  - Defined in <sys/socket.h>
  - Used in the first argument to the socket() function.

IPv6 Address Structure

- A new in6_addr structure holds a single IPv6 address:

```c
struct in6_addr {
  uint8_t s6_addr[16]; /* IPv6 address */
};
```

IPv4 Address Structure

- A new in_addr structure holds a single IPv4 address:

```c
struct in_addr {
  u_long s_addr;
};
```

Socket Address Structure

- New sockaddr_in6 structure holds IPv6 addresses (<netinet/in.h>)

```c
struct sockaddr_in6 {
  sa_family_t sin6_family; /* AF_INET6 */
  in_port_t sin6_port; /* transport layer port */
  uint32_t sin6_flowinfo; /* IPv6 traffic class & flow info */
  struct in6_addr sin6_addr; /* IPv6 address */
  uint32_t sin6_scope_id; /* set of intf. for a scope */
};
```

IPv6 functions:

- `sin6_flowinfo` contains the traffic class and the flow label.
- `sin6_scope_id` identifies a set of interfaces as appropriate for the scope of the address carried in the `sin6_addr` field.
  - Link scope: interface index.
  - Site scope: site identifier.
  - Not completely specified

IPv4 functions:

```c
struct sockaddr_in {
  short sin_family;
  u_short sin_port;
  struct in_addr sin_addr;
  char sin_zero[8];
};
```

The Socket Functions

- Applications call the socket() function to create a socket descriptor that represents a communication endpoint.

IPv6:

```c
s = socket(PF_INET6, SOCK_STREAM, 0); /* TCP Socket */
s = socket(PF_INET6, SOCK_DGRAM, 0); /* UDP Socket */
```

Once the application has created a PF_INET6 socket, it must use the sockaddr_in6 address structure when passing addresses in to the system.

- bind()
- connect()
- sendmsg()
- sendto()
Socket Interface Extensions for IPv6
The Socket Functions
- The system will use the sockaddr_in6 address structure to return addresses to applications that are using PF_INET6 sockets.
- The functions that return an address from the system to an application are:
  - accept()
  - recvfrom()
  - recvmsg()
  - getpeername()
  - getssockopt()
- No changes to the syntax of the socket functions are needed to support IPv6.

Compatibility with IPv4 Nodes
IPv6 applications are able to interoperate with IPv4 applications.
- Uses the IPv4-mapped IPv6 address format.
  - IPv4-mapped addresses are written as:
    ::FFFF:<IPv4-address>
- Applications may use PF_INET6 sockets to:
  - open TCP connections to IPv4 nodes
  - send UDP packets to IPv4 nodes
  - Encoding the destination's IPv4 address as an IPv4-mapped IPv6 address.
- When applications use PF_INET6 sockets to:
  - accept TCP connections from IPv4 nodes
  - receive UDP packets from IPv4 nodes
  - The system returns the peer's address using a sockaddr_in6 structure encoded this way.

IPV6 Wildcard Address
- While the bind() function allows applications to select the source IP address of UDP packets and TCP connections, applications often want the system to select the source address for them.
- With IPv4, one specifies the address as the symbolic constant INADDR_ANY.
- In IPv6 an address cannot be used to initialize an IPv6 address variable, but cannot be used in an assignment.
- Systems provide the wildcard in two forms:
  - extern const struct in6_addr in6addr_any;
  - struct in6_addr anyaddr = IN6ADDR_ANY_INIT;
  - Applications use in6addr_any similarly to the way they use INADDR_ANY in IPv4.

IPV6 Loopback Address
- Applications may need to send UDP packets to, or originate TCP connections to, services residing on the local node.
- In IPv4, they can do this by using the constant IPv4 address INADDR_LOOPBACK.
- The IPv6 loopback address is provided in two forms:
  - extern const struct in6_addr in6addr_loopback;
  - struct in6_addr loopbackaddr = IN6ADDR_LOOPBACK_INIT;
  - (can be used ONLY at declaration time)

Unicast Hop Limit
- A new setsockopt() option controls the hop limit used in outgoing unicast IPv6 packets.
- The name of this option is IPV6_UNICAST_HOPS, and it is used at the IPPROTO_IPV6 layer.
- Example:
  ```c
  int hoplimit = 10;
  if (setsockopt(s, IPPROTO_IPV6, IPV6_UNICAST_HOPS,
                 (char *) &hoplimit, sizeof(hoplimit)) == -1)
      perror("setsockopt IPV6_UNICAST_HOPS");
  ```

Sending and Receiving Multicast Packets
- IPv6 applications may send UDP multicast packets by simply specifying an IPv6 multicast address in the address argument of the sendto() function.
- Three socket options at the IPPROTO_IPV6 layer control some of the parameters for sending multicast packets.
  - IPV6_MULTICAST_IF: Set the interface to use for outgoing multicast packets. The argument is the index of the interface to use.
  - IPV6_MULTICAST_HOPS: Set the hop limit to use for outgoing multicast packets.
  - IPV6_MULTICAST_LOOP: If a multicast datagram is sent to a group to which the sending host itself belongs: 1: loop back a copy, 0: don't loop back a copy.
  - IPV6_JOIN_GROUP: Join a multicast group on a specified local interface.
  - IPV6_LEAVE_GROUP: Leave a multicast group on a specified interface.