

**IPv6**

Refs: Chapter 10, Appendix A

Netprog 2000 - IPv6 1

---

---

---

---

---

---

---

---

**IPv6 availability**

- Generally not part of O.S.
- Available in beta for many operating systems.
- Supposedly a complete implementation in FreeBSD.

Netprog 2000 - IPv6 2

---

---

---

---

---

---

---

---

**IPv6 Design Issues**

- Overcome IPv4 scaling problem
  - lack of address space.
- Flexible transition mechanism.
- New routing capabilities.
- Quality of service.
- Security.
- Ability to add features in the future.

Netprog 2000 - IPv6 3

---

---

---

---

---

---

---

---

## IPv6 Headers

- Simpler header - faster processing by routers.
  - No optional fields - fixed size (40 bytes)
  - No fragmentation fields.
  - No checksum
- Support for multiple headers
  - more flexible than simple “protocol” field.

Netprog 2000 - IPv6

4

---

---

---

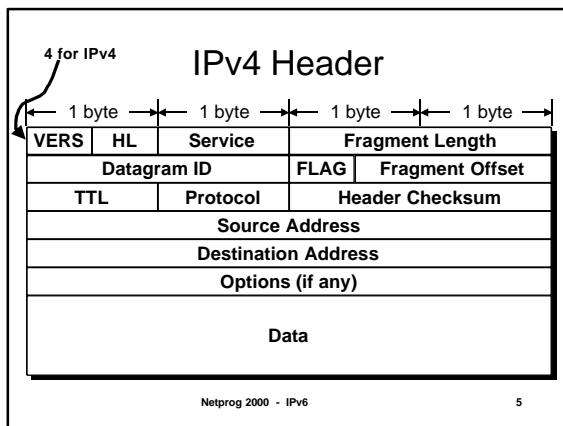
---

---

---

---

---



5

---

---

---

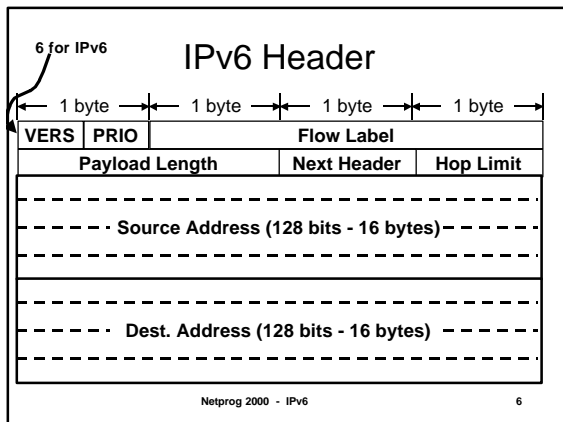
---

---

---

---

---



6

---

---

---

---

---

---

---

---

## IPv6 Header Fields

- VERS: 6 (IP version number)
- Priority: will be used in congestion control
- Flow Label: experimental - sender can label a sequence of packets as being in the same flow.
- Payload Length: number of bytes in everything following the 40 byte header, or 0 for a *Jumbogram*.

Netprog 2000 - IPv6

7

---

---

---

---

---

---

---

---

## IPv6 Header Fields

- Next Header is similar to the IPv4 "protocol" field - indicates what type of header follows the IPv6 header.
- Hop Limit is similar to the IPv4 TTL field (but now it really means hops, not time).

Netprog 2000 - IPv6

8

---

---

---

---

---

---

---

---

## Extension Headers

- Routing Header - source routing
- Fragmentation Header - supports fragmentation of IPv6 datagrams.
- Authentication Header
- Encapsulating Security Payload Header

Netprog 2000 - IPv6

9

---

---

---

---

---

---

---

---

## IPv6 Addresses

- 128 bits - written as eight 16-bit hex numbers.  
`5f1b:df00:ce3e:e200:0020:0800:2078:e3e3`
- High order bits determine the *type* of address. The book shows the breakdown of address types.

Netprog 2000 - IPv6

10

---

---

---

---

---

---

---

---

## IPv6 Aggregate Global Unicast Address

3	13	32	16	64
001	TLA ID	NLA ID	SLA ID	Interface ID

TLA: top-level aggregation  
NLA: next-level  
SLA: site-level

Interface ID is based on hardware MAC address

Netprog 2000 - IPv6

11

---

---

---

---

---

---

---

---

## IPv4-Mapped IPv6 Address

- IPv4-Mapped addresses allow a host that support both IPv4 and IPv6 to communicate with a host that supports only IPv4.
- The IPv6 address is based completely on the IPv4 address.

Netprog 2000 - IPv6

12

---

---

---

---

---

---

---

---

## IPv4-Mapped IPv6 Address

- 80 bits of 0s followed by 16 bits of ones, followed by a 32 bit IPv4 Address:

0000 . . . 0000	FFFF	IPv4 Address
80 bits	16 bits	32 bits

Netprog 2000 - IPv6

13

---

---

---

---

---

---

---

---

## Works with DNS

- An IPv6 application asks DNS for the address of a host, but the host only has an IPv4 address.
- DNS creates the IPv4-Mapped IPv6 address automatically.
- Kernel understands this is special address and really uses IPv4 communication.

Netprog 2000 - IPv6

14

---

---

---

---

---

---

---

---

## IPv4-Compatible IPv6 Address

- An IPv4 compatible address allows a host supporting IPv6 to talk IPv6 even if the local router(s) don't talk IPv6.
- IPv4 compatible addresses tell endpoint software to create a tunnel by encapsulating the IPv6 packet in an IPv4 packet.

Netprog 2000 - IPv6

15

---

---

---

---

---

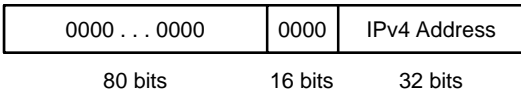
---

---

---

## IPv4-Compatible IPv6 Address

- 80 bits of 0s followed by 16 bits of 0s, followed by a 32 bit IPv4 Address:



Netprog 2000 - IPv6

16

---

---

---

---

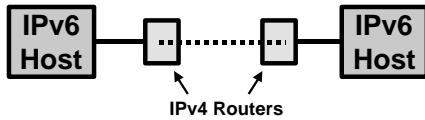
---

---

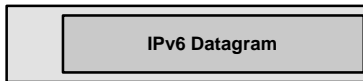
---

---

Tunneling  
(done automatically by kernel when IPv4-Compatible IPv6 addresses used)



IPv4 Datagram



Netprog 2000 - IPv6

17

---

---

---

---

---

---

---

---

## IPv6 Sockets programming

- New address family: `AF_INET6`
- New address data type: `in6_addr`
- New address structure: `sockaddr_in6`

Netprog 2000 - IPv6

18

---

---

---

---

---

---

---

---

## in6\_addr

```
struct in6_addr {  
    uint8_t s6_addr[16];  
};
```

Netprog 2000 - IPv6

19

---

---

---

---

---

---

---

---

## sockaddr\_in6

```
struct sockaddr_in6 {  
    uint8_t        sin6_len;  
    sa_family_t    sin6_family;  
    in_port_t      sin6_port;  
    uint32_t       sin6_flowinfo;  
    struct in6_addr sin6_addr;  
};
```

Netprog 2000 - IPv6

20

---

---

---

---

---

---

---

---

## Dual Server

- In the future it will be important to create servers that handle both IPv4 and IPv6.
- The work is handled by the O.S. (which contains protocol stacks for both v4 and v6):
  - automatic creation of IPv6 address from an IPv4 client (IPv4-mapped IPv6 address).

Netprog 2000 - IPv6

21

---

---

---

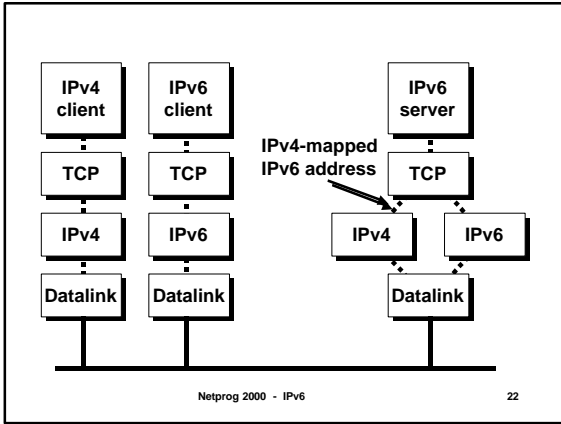
---

---

---

---

---




---

---

---

---

---

---

---

---

### IPv6 Clients

- If an IPv6 client specifies an IPv4 address for the server, the kernel detects and talks IPv4 to the server.
- DNS support for IPv6 addresses can make everything work.
  - `gethostbyname()` returns an IPv4 mapped IPv6 address for hosts that only support IPv4.

Netprog 2000 - IPv6 23

---

---

---

---

---

---

---

---

### IPv6 - IPv4 Programming

- The kernel does the work, we can assume we are talking IPv6 to everyone!
- In case we really want to know, there are some macros that determine the type of an IPv6 address.
  - We can find out if we are talking to an IPv4 client or server by checking whether the address is an IPv4 mapped address.

Netprog 2000 - IPv6 24

---

---

---

---

---

---

---

---