VNiVERSiDAD D SALAMANCA

http://informatica.usal.es/gii

Desarrollo de aplicaciones *Multicast*

Sockets en IPv6

Sockets *Multicast*

02/05/2017

VNiVERSiDAD D SALAMANCA

Sockets en IPv6: estructuras

- Estructura de direcciones de *sockets* en IPv6 (definidas en <netinet/in.h>)

```
/* IPv6 address */
struct in6_addr  {
   union {
     uint8_t u6_addr8[16];
     uint16_t u6_addr16[8];
     uint32_t u6_addr32[4];
   } in6_u;
};

struct sockaddr_in6  {
   uint8_t sin6_len;          /* Longitud de esta estructura*/
   sa_family_t sin6_family;   /* AF_INET6*/
   in_port_t sin6_port;       /* Transport layer port # */
   uint32_t sin6_flowinfo;    /* IPv6 flow information */
   struct in6_addr sin6_addr;/* IPv6 address */
   uint32_t sin6_scope_id;    /* IPv6 scope-id */
};
```


Sockets en IPv6: crear y asociar

```
/* Declaración */
int s;
struct sockaddr_in6 myaddr_in6;

/* Creación */
s = socket(AF_INET6, SOCK_STREAM, 0);

/* Inicialización */
myaddr_in6.sin6_family = AF_INET6;
myaddr_in6.sin6_addr   = in6addr_any;
myaddr_in6.sin6_port   = htons(13);

/* Asociación */
bind(s, (struct sockaddr *) &myaddr_in6, sizeof(myaddr_in6));
```


Sockets en IPv6: cambios de formato

- Cambios de formato en direcciones (<arpa/inet.h>)
  - `int inet_pton(int af, const char *src, void *dst);`
    - Convierte direcciones IPv4 (AF_INET) e IPv6 (AF_INET6) de texto a formato binario (estructura de dirección de red)
  - `const char *inet_ntop(int af, const void *src, char *dst, socklen_t size);`
    - Convierte direcciones IPv4 e IPv6 de formato binario a texto
- Ejemplo:

```
char equipo[INET6_ADDRSTRLEN];
unsigned char cliaddr[sizeof(struct in6_addr)];
inet_ntop (AF_INET6, cliaddr, equipo, sizeof(equipo));
```

- Ejemplo de sockets IPv6 de fecha y hora disponible en Diaweb

Otras familias y tipos de sockets (I)

Familia	Descripción
AF_INET	Protocolos IPv4
AF_INET6	Protocolos IPv6
AF_LOCAL (AF_UNIX)	Protocolos de dominios de UNIX
AF_ROUTE	Sockets de encaminamiento
AF_KEY	Sockets de clave
AF_NS	Protocolos Xerox NS (XNS)
AF_ISO	Protocolos de OSI

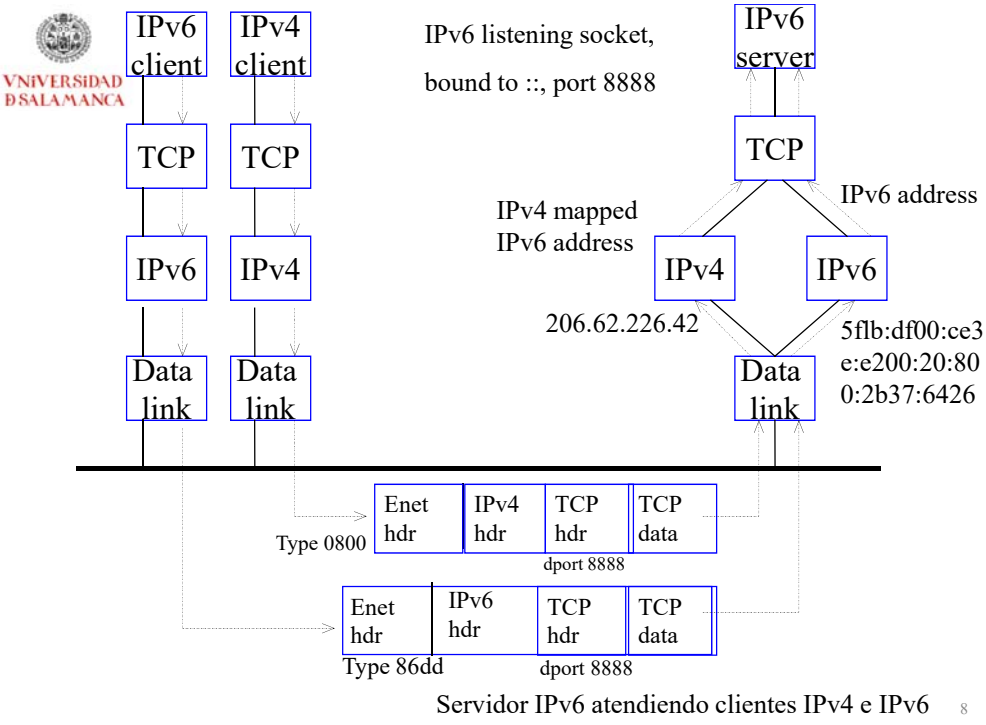
Tipo	Descripción
SOCK_STREAM	Sockets stream
SOCK_DGRAM	Sockets datagrama
SOCK_RAW	Procesado propio
SOCK_PACKET	Acceso a nivel de enlace (sólo para LINUX)
SOCK_SEQPACKET	Sockets de paquetes en secuencia

Otras familias y tipos de sockets (y II)

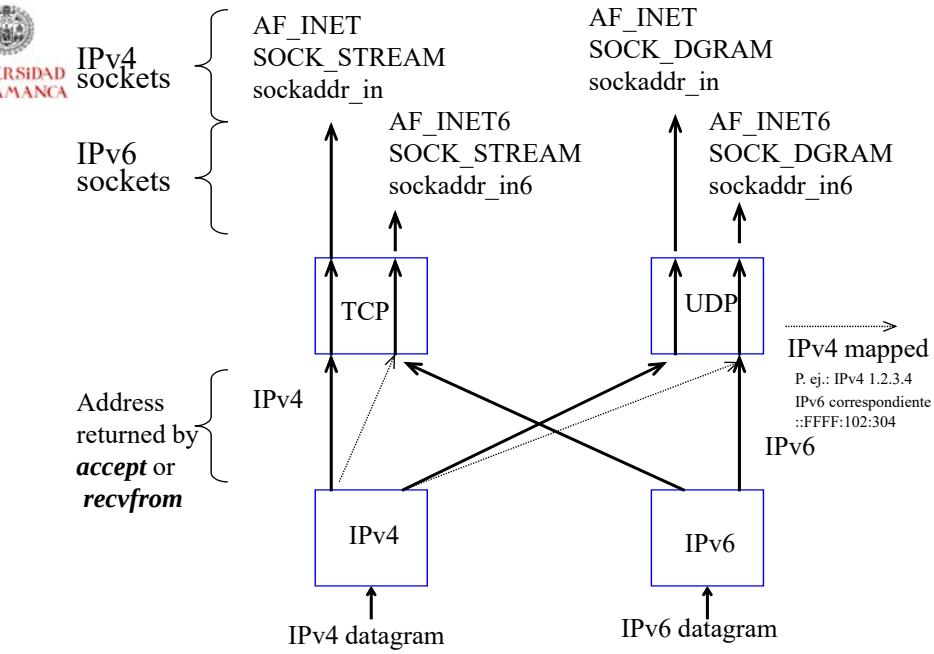
	AF_INET	AF_INET6	AF_LOCAL	AF_ROUTE	AF_KEY	AF_OSI	AF_NS
SOCK_STREAM	TCP	TCP	SI				
SOCK_DGRAM	UDP	UDP	SI				
SOCK_RAW	IPv4	IPv6		SI	SI		
SOCK_SEQPACKET						SI	SI

Escenario 1: Cliente IPv4, Servidor IPv6

- Un servidor IPv6 con pila dual puede atender clientes IPv4 e IPv6
- Lo hace utilizando direcciones IPv6 obtenidas de la correspondiente IPv4
- El servidor crea un socket IPv6 que escucha en la dirección comodín de IPv6



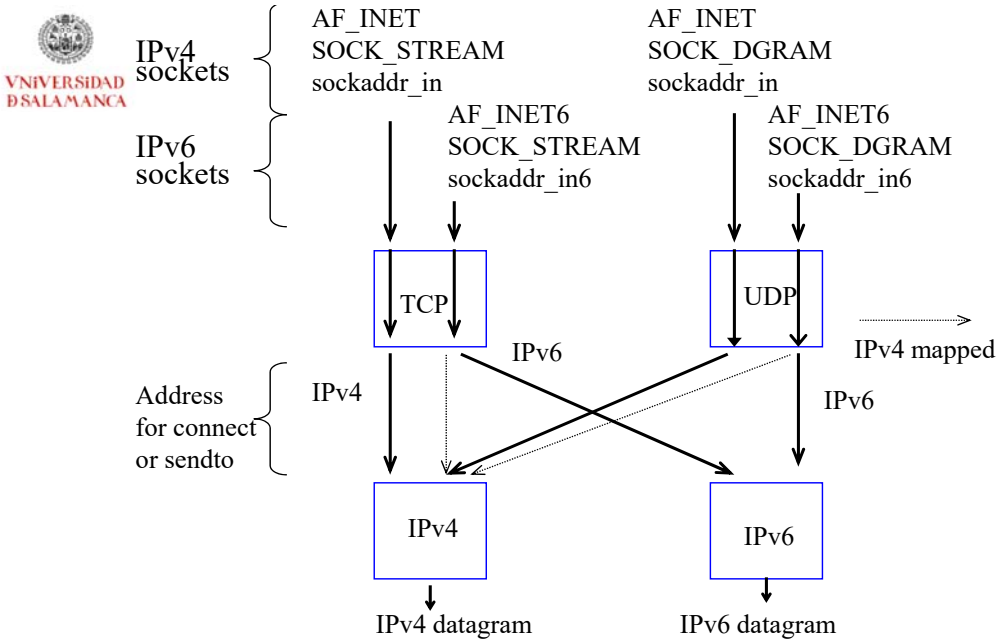
Servidor IPv6 atendiendo clientes IPv4 e IPv6 8



Recepción de datagramas, dependiendo del tipo de *socket* receptor

Escenario 2: Cliente IPv6, Servidor IPv4

- El servidor IPv4 se inicia en un nodo que sólo tiene IPv4 y crea un sockets IPv4 de escucha
- El cliente IPv6 se inicia y antes del *connect* o *sendto* llama a ***getaddrinfo*** para obtener la dirección IP del servidor. **La pila dual del cliente detecta la dirección IPv4 del servidor y retorna la dirección IPv6 de la correspondiente IPv4**
- Por tanto, se hace utilizando datagramas IPv4



Peticiones de clientes, dependiendo del tipo de dirección y tipo de socket

Ejemplo

- Servidor y cliente de fecha y hora TCP y UDP disponible en la plataforma de la asignatura

Aplicaciones *multicast* con Sockets

- *Multicast*
  - Mediante las funciones `setsockopt` y `getsockopt` con las siguientes opciones

Orden	Tipo de dato	Descripción
`IP_ADD_MEMBERSHIP` `IPV6_ADD_MEMBERSHIP`	`struct ip_mreq` `struct ipv6_mreq`	`Unirse a un grupo multicast`
`IP_DROP_MEMBERSHIP` `IPV6_DROP_MEMBERSHIP`	`struct ip_mreq` `struct ipv6_mreq`	`Abandonar un grupo multicast`
`IP_MULTICAST_IF` `IPV6_MULTICAST_IF`	`struct in_addr`	`Especificar el interfaz por defecto`
`IP_MULTICAST_TTL` `IPV6_MULTICAST_HOPS`	`u_char`	`Especificar el TTL en IPv4 y el alcance para IPv6`
`IP_MULTICAST_LOOP` `IPV6_MULTICAST_LOOP`	`u_char`	`Habilitar o deshabilitar loopback`

Aplicaciones *multicast* con Sockets (I)

- Emisor *multicast* (Cliente)
  - Crear el socket AF_INET (ó AF_INET6) y del tipo SOCK_DGRAM
  - Inicializar la estructura sockaddr_in (ó sockaddr_in6) con nuestra propia IP y número de puerto efímero
  - Activar IP_MULTICAST_LOOP (ó IPV6_MULTICAST_LOOP) en el socket si se desea recibir también copia de los datagramas enviados al grupo *multicast*
  - Configurar si fuera necesario la interfaz por la que se enviarán los datagramas con la opción IP_MULTICAST_IF (ó IPV6_MULTICAST_IF)
  - Enviar el datagrama a la IP *multicast* y número de puerto de nuestra aplicación (conocido por nuestros clientes) rellenando una estructura sockaddr_in (ó sockaddr_in6)
- Suscriptor *multicast* (Servidor)
  - Crear el socket AF_INET (ó AF_INET6) y del tipo SOCK_DGRAM
  - Activar la opción SO_REUSEADDR para permitir varios suscriptores en la misma máquina
    - De esta forma se pueden recibir datagramas destinados al mismo numero de puerto
  - Usar *bind* para especificar el número de puerto de nuestra aplicación e INADDR_ANY (ó in6addr_any) para recibir datagramas
  - Utilizar IP_ADD_MEMBERSHIP (ó IPV6_ADD_MEMBERSHIP) para especificar el grupo *multicast* del que se desean recibir datagramas y opcionalmente por qué interfaz si el equipo tiene varias (estructura *pv6_mreq*)
  - Recibir los datagramas

Aplicaciones *multicast* con Sockets (II)

- Extracto de código de unión al grupo *multicast* en una determinada interfaz

```
...
#define GRUPOMULTICASTV6 "ff15::33“
#define INTERFAZ “eth0”
...

struct ipv6_mreq ipv6mreq;
...
ipv6mreq.ipv6mr_interface=if_nametoindex(INTERFAZ);

/* Convierte la dirección multicast a binario */
if(inet_pton(familia,GRUPOMULTICASTV6,&ipv6mreq.ipv6mr_multiaddr)==-1)
   {
     perror(“Llamada inet_pton\n");
     exit(1);
   }

/* Unirse al grupo multicast */
if(setsockopt(s,IPPROTO_IPV6,IPV6_ADD_MEMBERSHIP,&ipv6mreq,sizeof(ipv6mreq))==-1)
   {
     perror(“Llamada setsockopt para multicast\n");
     exit(1);
   }
```


Aplicaciones *multicast* con Sockets (y III)

- Extracto de código para especificar la interfaz por la que difundir

```
...
#define INTERFAZ “eth0”
...

int interfaz;
...

interfaz=if_nametoindex(INTERFAZ);

/* Difundir por una interfaz determinada */
if(setsockopt(s,IPPROTO_IPV6,IPV6_MULTICAST_IF,(char *)&interfaz, sizeof(interfaz)))==-1)
   {
     perror(“Llamada setsockopt para especificar interfaz de difusión");
     exit(1);
   }
```


Aplicaciones *multicast* con Sockets (y III)

- Extracto de código para especificar el número de saltos

```
...
#define INTERFAZ “eth0”
...

int saltos;
...

Saltos=15;

if(setsockopt(s, IPPROTO_IPV6, IPV6_MULTICAST_HOPS, &saltos, sizeof(saltos)) == -1) {
                perror("Difusor: Error al especificar el nUmero de saltos\n");
                exit(-1);
                }
```