Portando aplicaciones a IPv6

Eva M. Castro – eva@gsyc.escet.urjc.es

Grupo de Sistemas y Comunicaciones (GSyC)
Departamento de Informática, Estadística y Telemática (DIET)
Universidad Rey Juan Carlos (URJC)

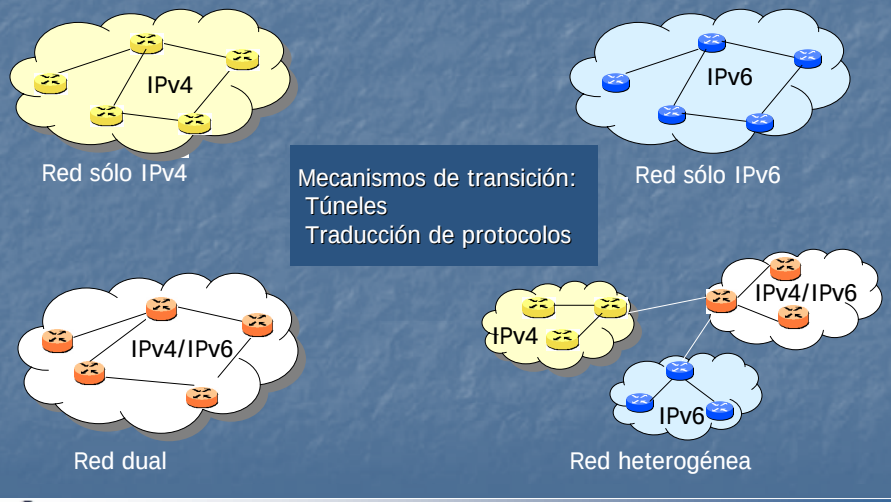
Agenda

- Arquitectura de transición
- Evolución de aplicaciones
- Escenarios de transición de aplicaciones
- Consideraciones al portar aplicaciones
- API de sockets BSD
- Aplicaciones independientes de la versión IP
- Recomendaciones

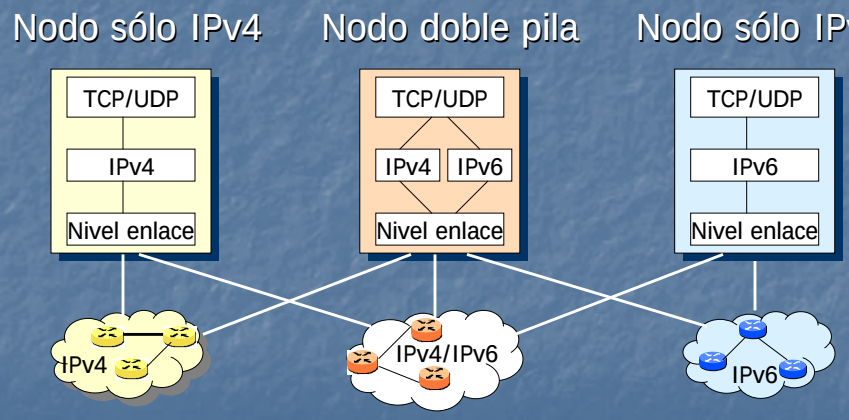
Arquitectura de transición

- Red
- Nodos terminales
- Aplicaciones

Red (encaminamiento / direccionamiento)

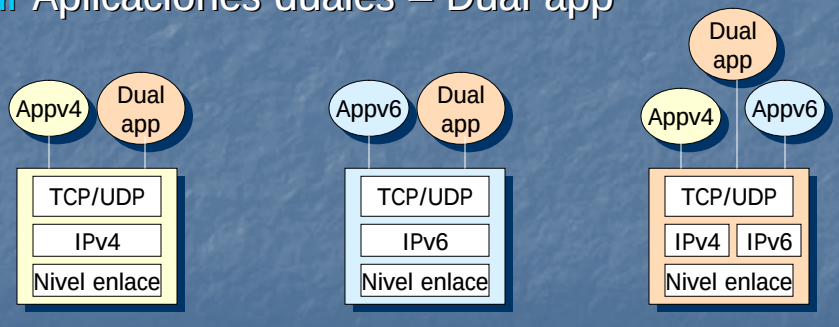


Nodos terminales (Pila IP)

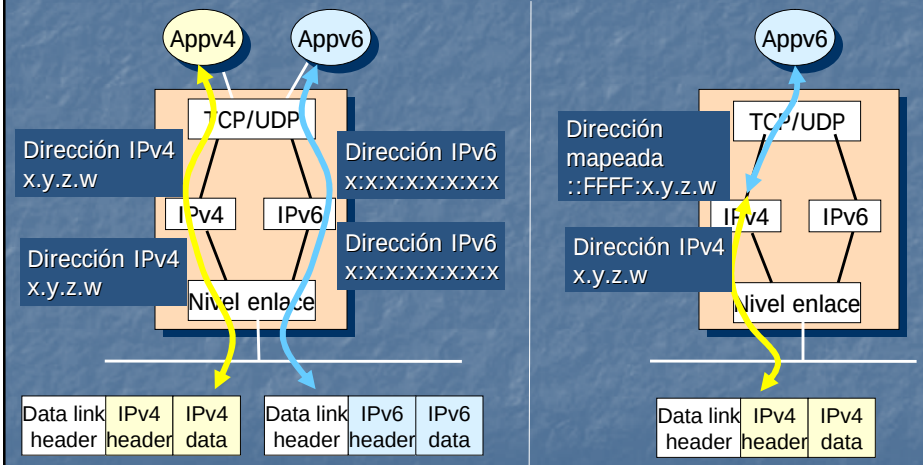


Aplicaciones (código fuente)

- Aplicaciones sólo IPv4 – Appv4
- Aplicaciones sólo IPv6 – Appv6
- Aplicaciones duales – Dual app



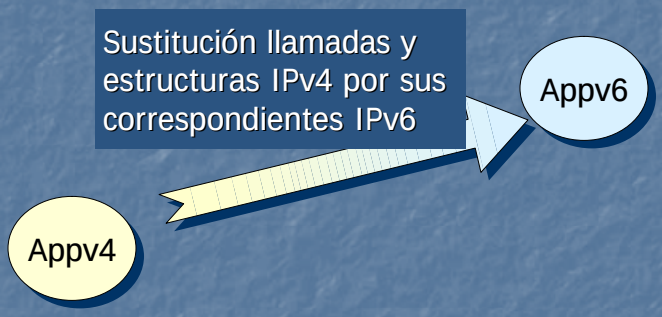
Aplicaciones – Nodos duales



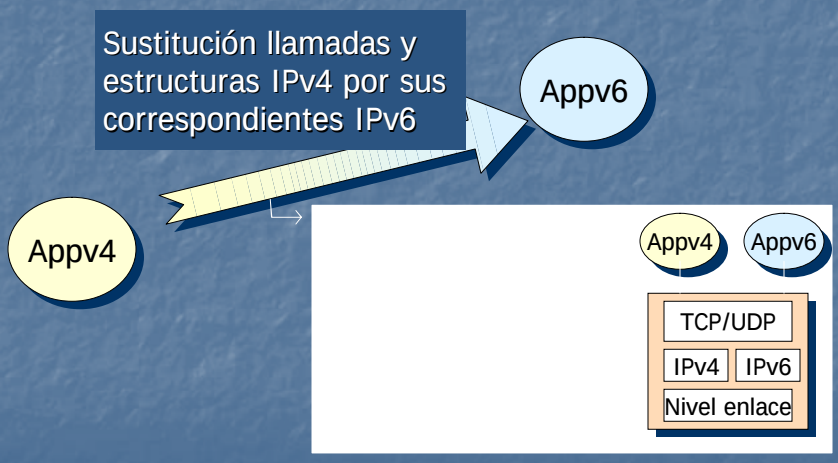
Agenda

- Arquitectura de transición
- Evolución de aplicaciones
- Escenarios de transición de aplicaciones
- Consideraciones al portar aplicaciones
- API de sockets BSD
- Aplicaciones independientes de la versión IP
- Recomendaciones

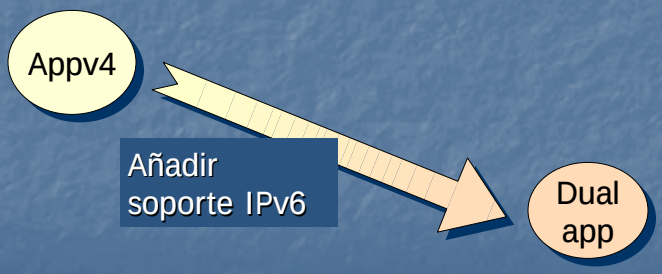
Evolución de las aplicaciones



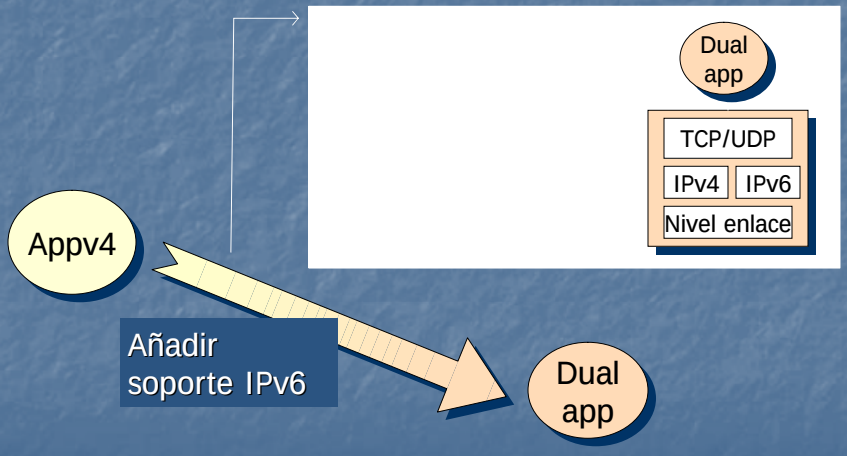
Evolución de las aplicaciones



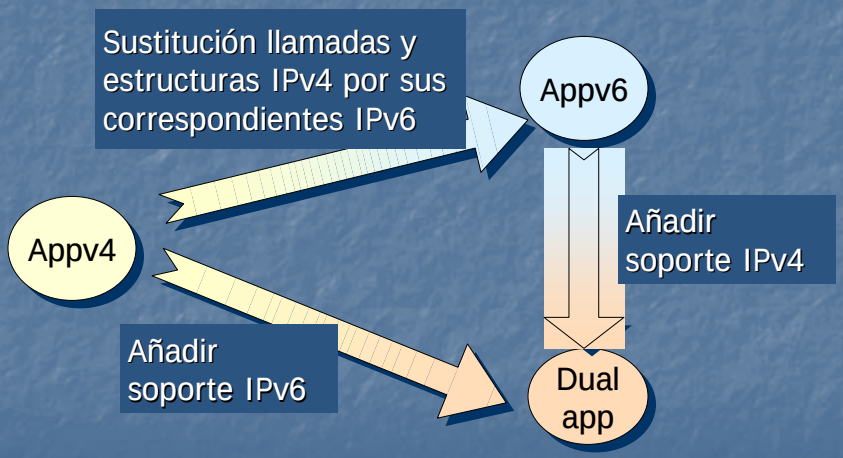
Evolución de las aplicaciones



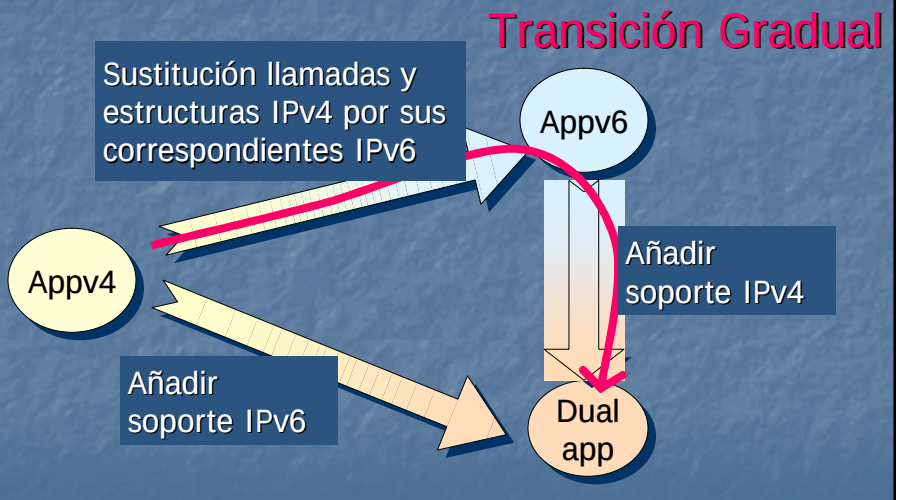
Evolución de las aplicaciones



Evolución de las aplicaciones



Evolución de las aplicaciones



Agenda

- Arquitectura de transición
- Evolución de aplicaciones
- Escenarios de transición de aplicaciones
- Consideraciones al portar aplicaciones
- API de sockets BSD
- Aplicaciones independientes de la versión IP
- Recomendaciones

Escenarios de transición de aplicaciones

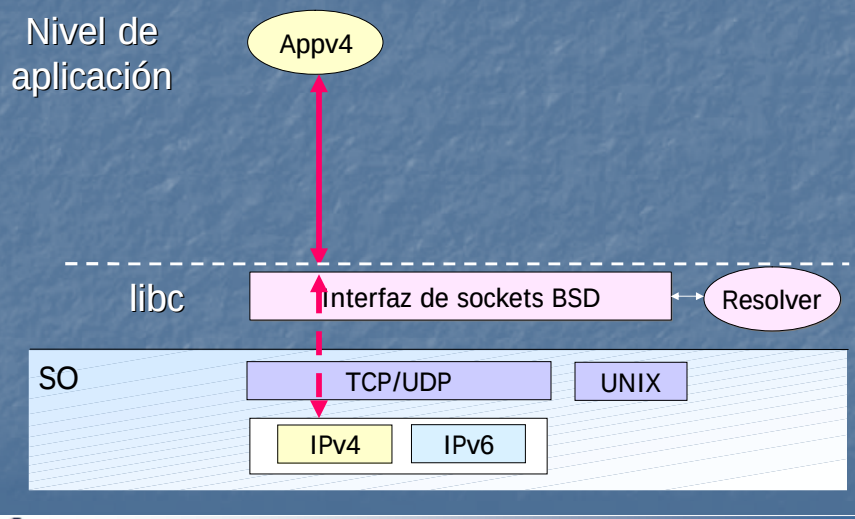
Partiendo de aplicaciones IPv4 que funcionan en nodos IPv4:

1. Aplicaciones IPv4 en nodos duales
2. Aplicaciones IPv6 en nodos duales
3. Aplicaciones duales en nodos duales
4. Aplicaciones duales en nodos IPv4

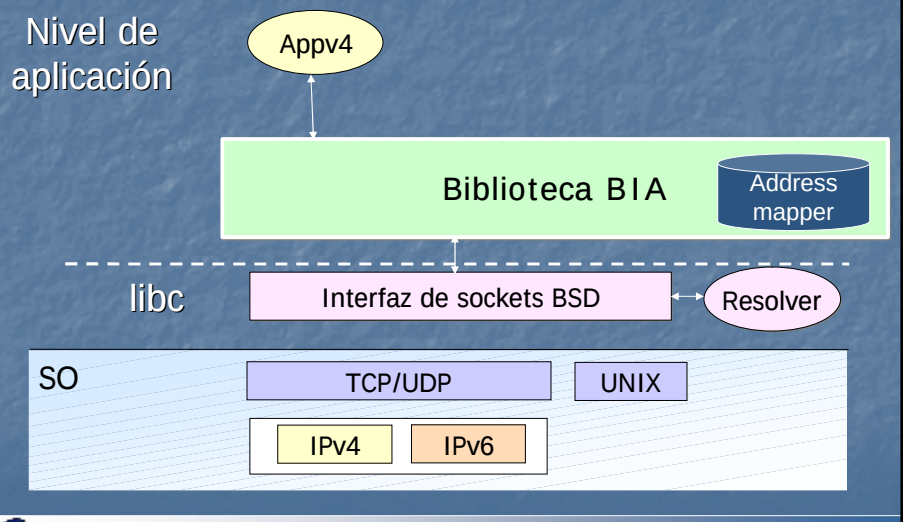
1. Aplicaciones IPv4 en nodos duales

- Código fuente con dependencias IPv4
- Intercambio de paquetes IPv4
- Para su funcionamiento en redes IPv6:
  - Portar el código a IPv6, las aplicaciones usan IPv6 de forma nativa, o
  - Utilizar mecanismos de transición. Las aplicaciones son IPv4 pero se intercambian paquetes IPv6:
    - **BIA**: Aplicaciones IPv4 + Nodos duales
    - **BIS**: Aplicaciones IPv4 + Nodos (duales/IPv4)

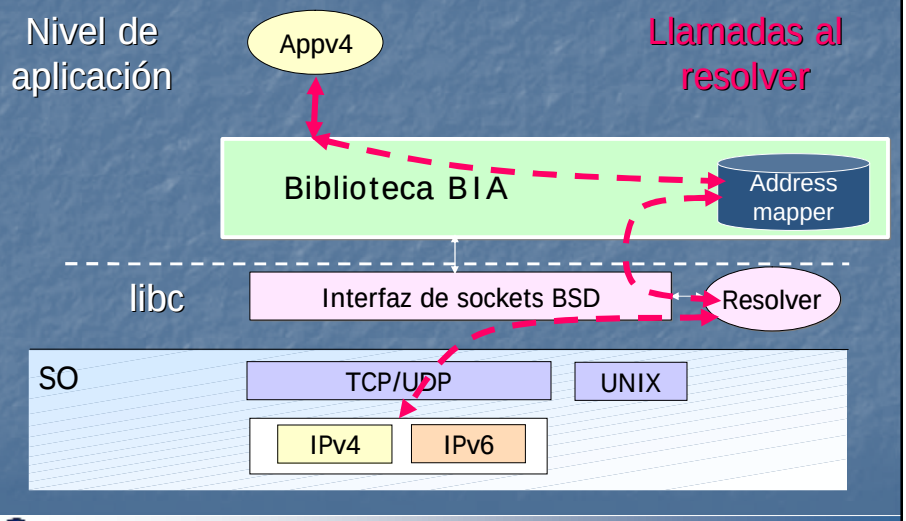
Aplicaciones IPv4



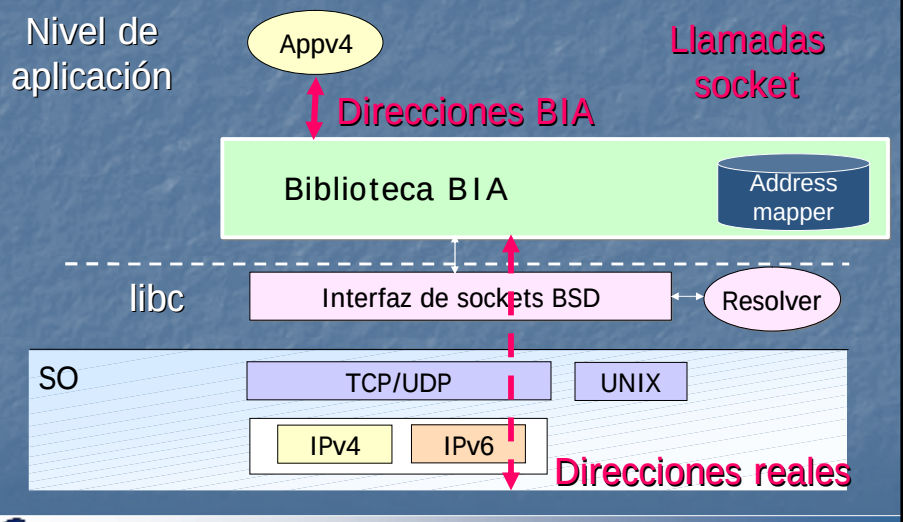
BIA – Bump In the API



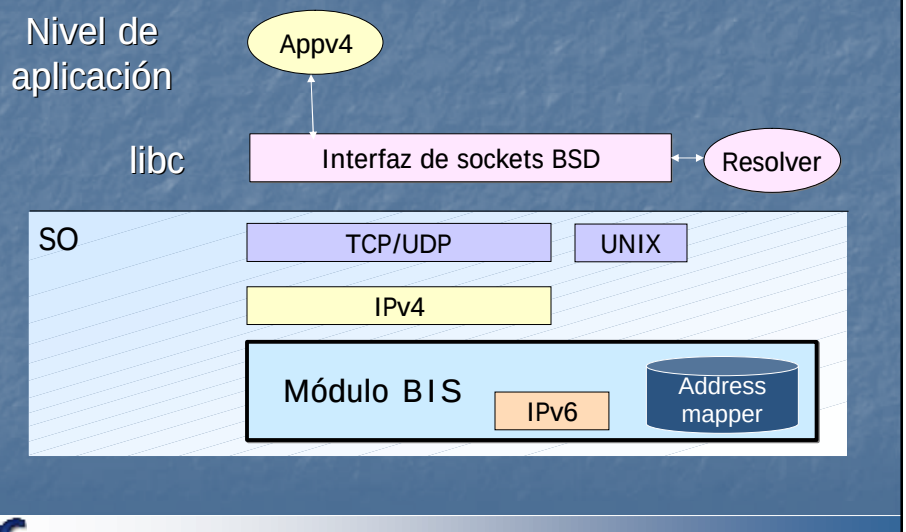
BIA – Llamadas al resolver



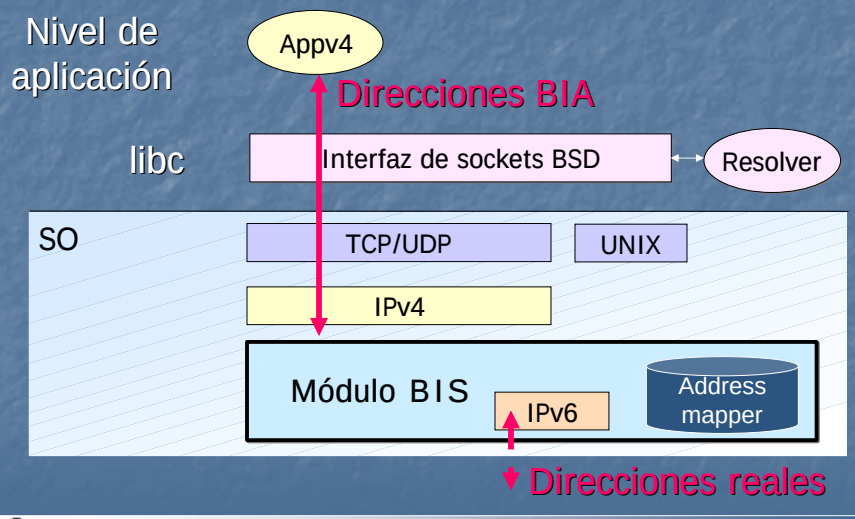
BIA – Llamadas socket



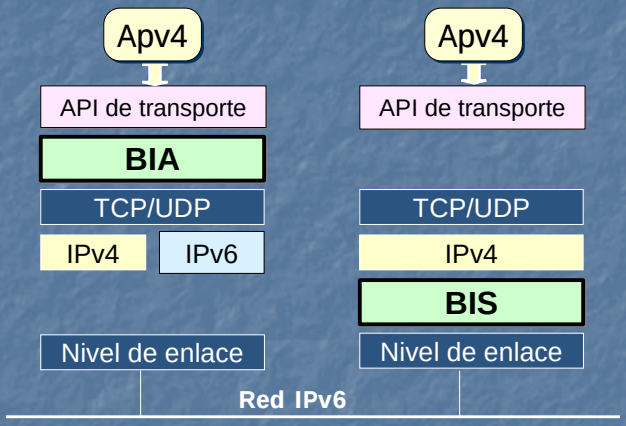
BIS – Bump In the Stack



BIS – Bump In the Stack

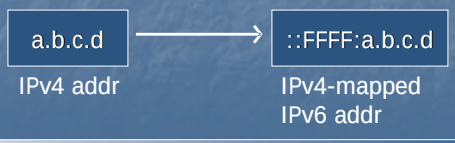


BIS/BIA

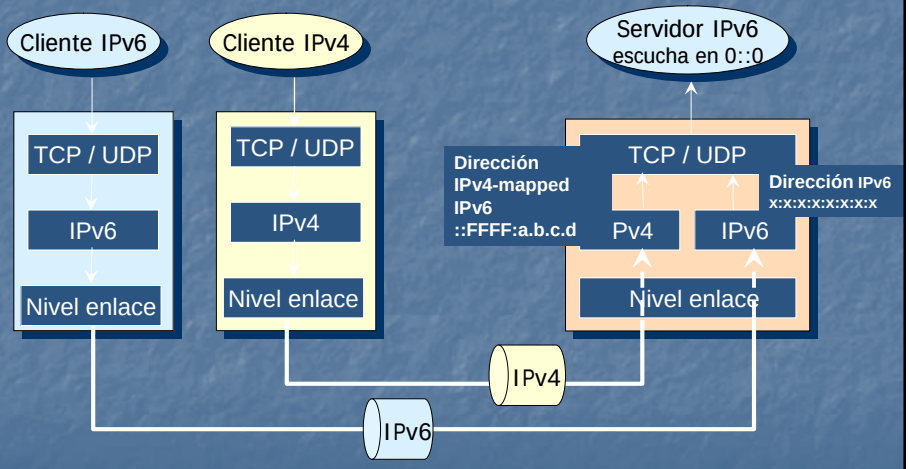


2. Aplicaciones IPv6 en nodos duales

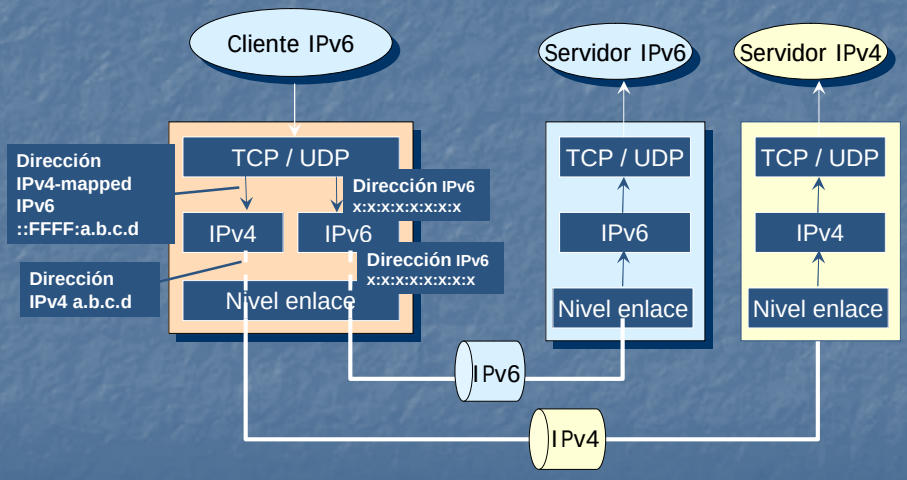
- Código fuente con dependencias IPv6
- Intercambio de paquetes IPv6
- Si la red es IPv4:
  - Convertir la aplicación en dual, o
  - Dos aplicaciones diferentes: ping4 y ping6, o
  - Utilizar direcciones IPv6 a partir de las direcciones IPv4, *IPv4-mapped IPv6 addresses*, no soportadas en todos las implementaciones



Aplicación servidor IPv6 en nodo dual



Aplicación cliente IPv6 en nodo dual



Interoperabilidad Cliente/Servidor

		Servidor IPv4		Servidor IPv6	
		Nodo IPv4	**Nodo Dual**	**Nodo IPv6**	**Nodo Dual**
Cliente IPv4	**Nodo IPv4**	**IPv4**	**IPv4**		
	Nodo Dual	**IPv4**	**IPv4**		
Servidor IPv6	Nodo IPv6			**IPv6**	**IPv6**
	Nodo Dual			**IPv6**	**IPv6**

Interoperabilidad Cliente/Servidor

		Servidor IPv4		Servidor IPv6	
		Nodo IPv4	**Nodo Dual**	**Nodo IPv6**	**Nodo Dual**
Cliente IPv4	**Nodo IPv4**	**IPv4**	**IPv4**	**Error**	**IPv4**
	Nodo Dual	**IPv4**	**IPv4**	**Error**	**IPv4**
Servidor IPv6	Nodo IPv6	**Error**	**Error**	**IPv6**	**IPv6**
	Nodo Dual	**IPv4**	**IPv4**	**IPv6**	**IPv6**

3. Aplicaciones duales en nodos duales

- Aplicaciones válidas para redes IPv4 e IPv6:
  - Implementación de **aplicaciones cliente**:
    - Resolver nombre de máquina del servidor a las posibles direcciones IP. Intentar conectar primero usando IPv6 y si falla probar con IPv4.
  - Implementaciones de **aplicaciones servidor**:
    1. Mantener conexiones diferentes de forma explícita para IPv4 e IPv6, o
    2. Desarrollar una aplicación servidor IPv6 y confiar en las direcciones IPv4-mapped IPv6 para los clientes IPv4.

4. Aplicaciones duales en nodos IPv4

- Las aplicaciones duales deberían funcionar en los nodos sólo IPv4 para evitar tener varias versiones de la misma aplicación.

REQUISITO

- Desarrollar el código fuente para que nodos que no tengan soporte del protocolo IPv6 puedan ejecutar dichas aplicaciones.

Agenda

- Arquitectura de transición
- Evolución de aplicaciones
- Escenarios de transición de aplicaciones
- Consideraciones al portar aplicaciones
- API de sockets BSD
- Aplicaciones independientes de la versión IP
- Recomendaciones

Consideraciones al portar aplicaciones

Dependencias con la versión IP dentro de las aplicaciones:

1. Formato de presentación de las direcciones IP
2. API del nivel de transporte
3. Resolución de nombres/direcciones
4. Dependencias específicas

1. Formato de presentación de las direcciones IP

El formato de presentación de las direcciones IP es una cadena de caracteres: "10.0.0.1"

PROBLEMAS

- Memoria necesaria para el formato de presentación de una dirección IPv4 es menor que para una IPv6.
- IPv4 usa "." como separador, IPv6 uses ":". Los analizadores de direcciones deben soportar ambos.
- Ambigüedad en el uso del carácter ":" en URLs: http://[DirecciónIPv6]:puerto

RECOMENDACIÓN

- Usar FQDN (Fully Qualified Domain Name)

2. API del nivel de transporte

- Estructuras de datos de red.
- Funciones de conversión de direcciones.
- Funciones del API de comunicaciones.
- Opciones de configuración de red.

PROBLEMAS
• El API de IPv4 hace visible a las aplicaciones dependencias con la versión de IP. Es necesario cambiar funciones y estructuras de de datos.

RECOMENDACIÓN
• Desarrollar aplicaciones independientes de la versión IP

3. Resolución de nombres/direcciones

- Resolución: directa e inversa.
- Peticiones/Respuestas de DNS se envían con IPv4/IPv6, independientemente del contenido de la consulta.

•PROBLEMAS
• Las funciones de resolución de nombres/direcciones IPv4 no son válidas para IPv6.

•RECOMENDACIÓN
• Utilizar funciones y estructuras de datos independientes de la versión IP, si el API lo proporciona.

4. Dependencias específicas

- Selección de dirección IP.
- Fragmentación a nivel de aplicación.
- Almacenamiento de direcciones IP.

PROBLEMAS
• Pueden existir más dependencias que las estructuras de datos y las llamadas al API de comunicaciones.

RECOMENDACIÓN
•Revisar exhaustivamente el código fuente.

Agenda

- Arquitectura de transición
- Evolución de aplicaciones
- Escenarios de transición de aplicaciones
- Consideraciones al portar aplicaciones
- API de sockets BSD
- Aplicaciones independientes de la versión IP
- Recomendaciones

API de sockets BSD

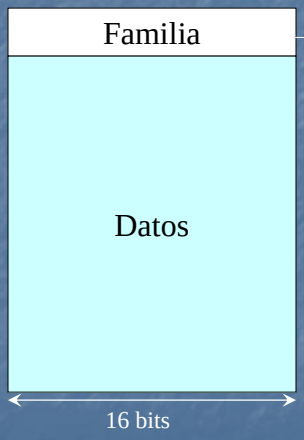
La longitud de las direcciones IPv6 es 128 bits .

Cambios en la parte de comunicaciones de las aplicaciones

1. Estructuras de datos:
   Estructura de dirección de socket
2. Funciones de conversión de direcciones:
   - Cadena de caracteres y estructura
   - Nombre y estructura
3. Funciones de comunicación

Dirección de socket genérica

sockaddr

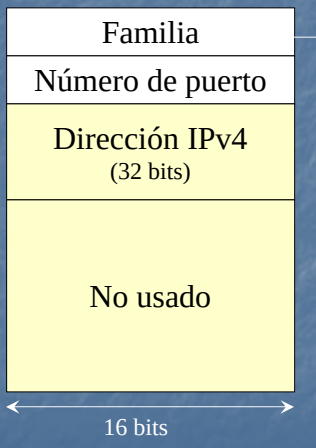


AF_XXXX

```
struct sockaddr {
   sa_family_t  sa_family;
   char         sa_data[14];
};
```

Estructura de dirección de socket IPv4

sockaddr_in



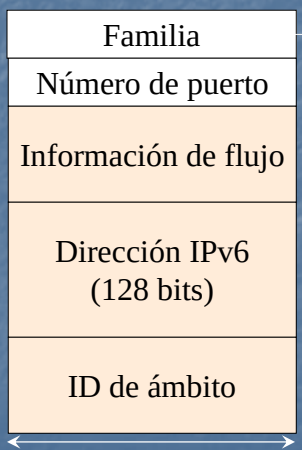
AF_INET

```
struct sockaddr_in {
   sa_family_t     sin_family;
   in_port_t       sin_port;
   struct in_addr  sin_addr;
   char            sin_zero[8];
};

struct in_addr {
   uint32_t        s_addr;
};
```


Estructura de dirección de socket IPv6

sockaddr_in6



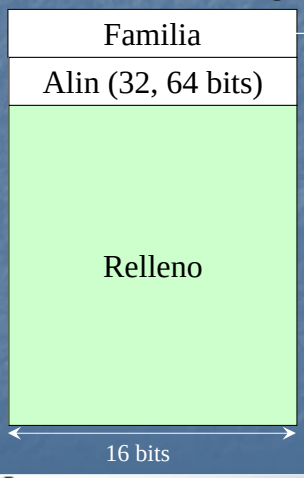
AF_INET6

```
struct sockaddr_in6 {
   sa_family_t      sin6_family;
   in_port_t        sin6_port;
   uint32_t         sin6_flowinfo;
   struct in6_addr  sin6_addr;
   uint32_t         sin6_scope_id;
};

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

Estructura independiente de protocolo

sockaddr_storage



AF_XXXX

```
struct sockaddr_storage {
   sa_family_t      sin6_family;
    __ss_aligntype __ss_align;
    char __ss_padding[_SS_PADSIZE];
};
```


Reserva de estructuras

Sólo IPv4

```
struct sockaddr_in serverAddr;
/* … */
bind(serverfd, (struct sockaddr *)&serverAddr, &alen);
```

Sólo IPv6

```
struct sockaddr_in6 serverAddr;
/* … */
bind(serverfd, (struct sockaddr *)&serverAddr, &alen);
```

Independiente de Protocolo

```
struct sockaddr_storage serverAddr;
/* … */
bind(serverfd, (struct sockaddr *)&serverAddr, &alen);
```

API de sockets BSD

La longitud de las direcciones IPv6 es 128 bits .

Cambios en la parte de comunicaciones de las aplicaciones

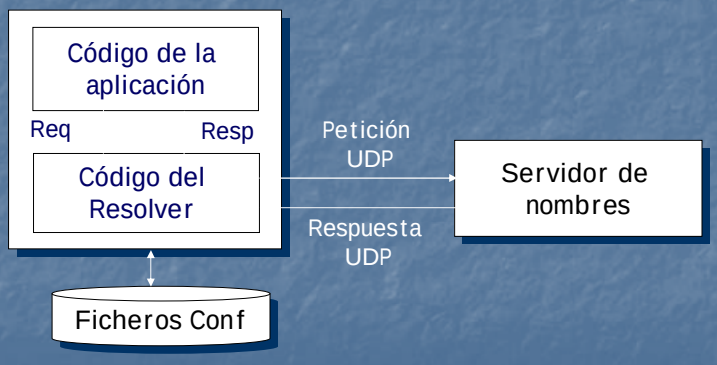
1. Estructuras de datos:
   Estructura de dirección de socket

2. Funciones de conversión de direcciones:
   - Cadena de caracteres y estructura
   - Nombre y estructura

3. Funciones de comunicación

Funciones de conversión

- RESOLVER: Devuelve la dirección IP asociada a un nombre (o a la inversa).



Conversión: nombre & dirección IP

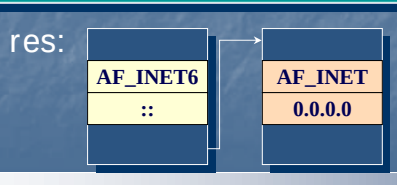
Sólo IPv4 (direcciones v4)	IPv4 & IPv6 (direcciones v4, v6, v4-map v6)
`gethostbyname()`	`getaddrinfo()`
`gethostbyaddr()`	`getnameinfo()`

Funciones idependientes de protocolo

Getaddrinfo

```
getaddrinfo(NULL, DAYTIME_PORT, &hints, &res);

freeaddrinfo(res);
```



Getnameinfo

```
getnameinfo((struct sockaddr *)&clientAddr, addrLen,
            clientHost, sizeof(clientHost),
            clientPort, sizeof(clientPort),
            NI_NUMERICHOST);
```


Conversión: cadena & estructura

- Conversión entre el formato de presentación (IPv4: “10.0.0.1”) y la estructura de datos correspondiente (IPv4: sockaddr_in)

	Sólo IPv4	IPv4 & IPv6
Cadena -> Estructura	`inet_aton()` `inet_addr()`	`inet_pton()`
Estructura -> Cadena	`inet_ntoa()`	`inet_ntop()`

inet_pton

Sólo IPv4

```
inet_aton(addrStr4,
          &addr4);
```

Sólo IPv6

```
inet_pton(AF_INET6, addrStr6,
          &addr6);
```

IPv4/IPv6

```
inet_pton(family, addrStr, &addr);
```


inet_ntop

Sólo IPv4

```
addrStr4 = inet_ntoa(addr4);
```

Sólo IPv6

```
inet_ntop(AF_INET6, addr6,
          addrStr6,
          INET6_ADDRSTRLEN);
```

IPv4/IPv6

```
inet_ntop(addr.ss_family, addr, addrStr, sizeof(addrStr));
```

API de sockets BSD

La longitud de las direcciones IPv6 es 128 bits .

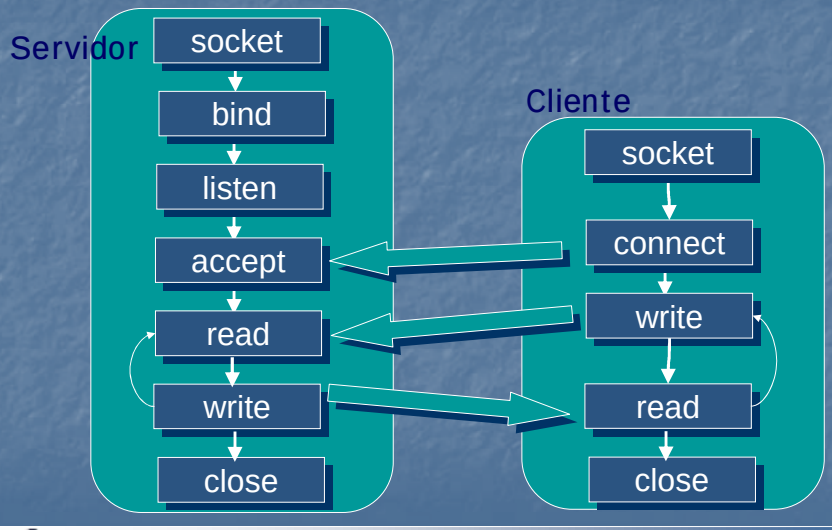
Cambios en la parte de comunicaciones de las aplicaciones

1. Estructuras de datos:
   Estructura de dirección de socket

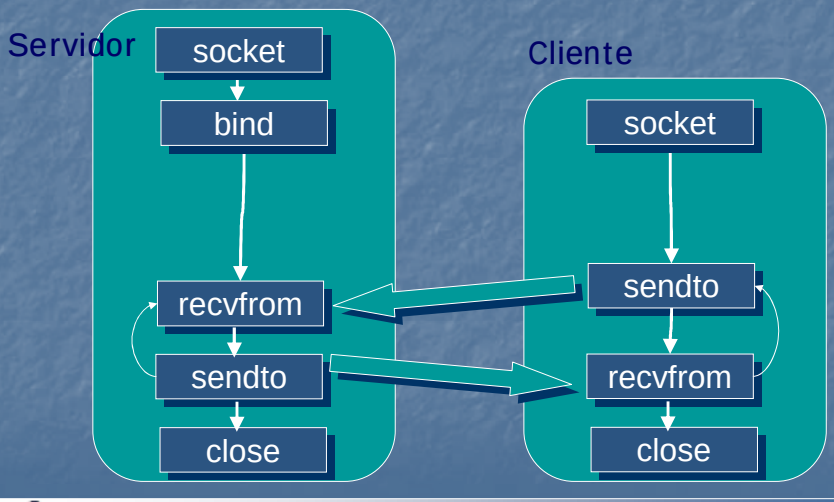
2. Funciones de conversión de direcciones:
   - Cadena de caracteres y estructura
   - Nombre y estructura

3. Funciones de comunicación

Funciones de comunicación (TCP)



Funciones de comunicación (UDP)



Mismas funciones en IPv4 & IPv6

- **Familia de direcciones, tipo de socket y protocolo en la función `socket`:**

```
int socket (int family, int type, int protocol);
```

- **Conversión de tipos de los diferentes tipos de estructura de socket a una estructura de socket genérica `struct sockaddr *`.**

IPv4: **desde** `(struct sockaddr_in *)`
IPv6: **desde** `(struct sockaddr_in6 *)`
Independiente de p**rotocolo:** **desde** `(struct sockaddr_storage *)`

- **Tamaño de la estructura de socket.**

Opciones de configuración (setsockopt)

- IPV6_UNICAST_HOPS
- Multicast:
  - IPV6_MULTICAST_IF
  - IPV6_MULTICAST_HOPS
  - IPV6_MULTICAST_LOOP
  - IPV6_JOIN_GROUP
  - IPV6_LEAVE_GROUP
- IPV6_V6ONLY for AF_INET6

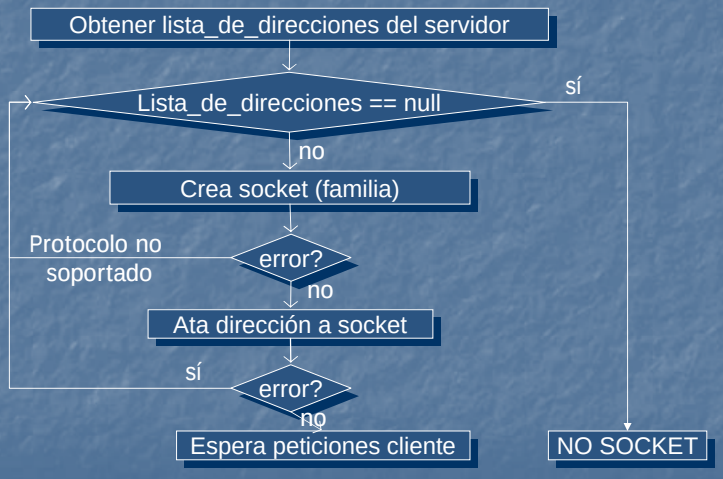
Agenda

- Arquitectura de transición
- Evolución de aplicaciones
- Escenarios de transición de aplicaciones
- Consideraciones al portar aplicaciones
- API de sockets BSD
- Aplicaciones independientes de la versión IP
- Recomendaciones

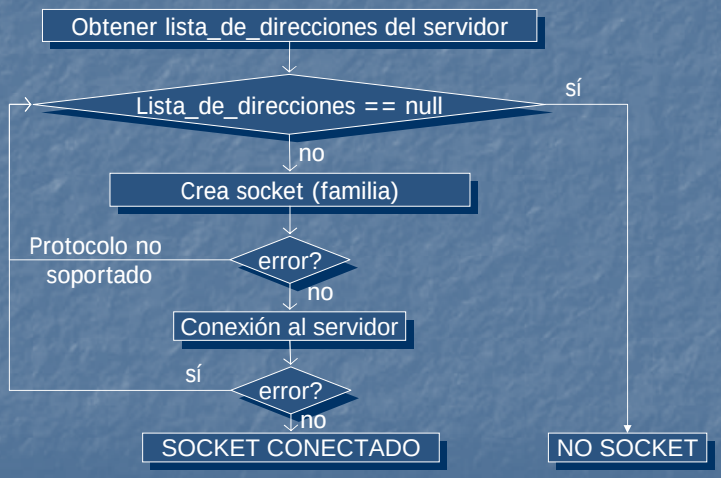
Aplicaciones independientes de la versión IP

- Usar estructuras de datos y funciones independientes de la versión IP:
  - sockaddr_storage
  - getaddrinfo()/getnameinfo()
- No usar inet_ntop()/inet_pton()
- Bucle para encontrar la dirección IP válida:
  - Clientes:
    - Conexión a una de las posibles direcciones IP del servidor, aquella en la que se pueda establecer la comunicación.

Ejemplo de aplicación servidor



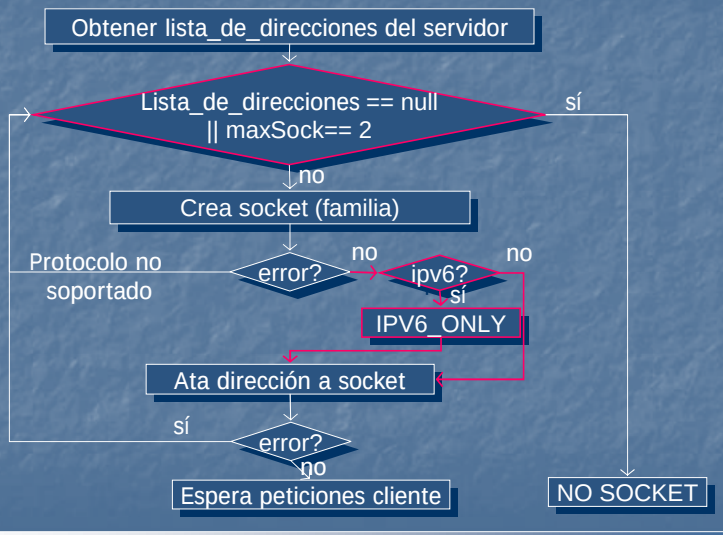
Ejemplo de aplicación cliente



Aplicaciones servidor duales

- Servidores IPv6:
  - Clientes IPv4 se conectan usando una dirección IPv4 que se convierte en (::FFFF:a.b.c.d).
  - Clientes IPv6 se conectan usando una dirección IPv6.
- Servidores duales: utilizan diferentes sockets IPv4 e IPv6 (IPV6_ONLY):
  - Clientes IPv4 se conectan usando una dirección IPv4
  - Clientes IPv6 se conectan usando una dirección IPv6

Servidores duales: IPv4 & IPv6 sockets



Agenda

- Arquitectura de transición
- Evolución de aplicaciones
- Escenarios de transición de aplicaciones
- Consideraciones al portar aplicaciones
- API de sockets BSD
- Aplicaciones independientes de la versión IP
- Recomendaciones

Recomendaciones para las aplicaciones

- Desarrollar aplicaciones duales:
  - Válidas para cualquier tipo de nodo y para comunicarse con cualquier aplicación utilizando IPv4 o IPv6.
  - Intentar la comunicación con cada una de las direcciones IP obtenidas a través de las funciones de resolución.
- Usar FQDN:
  - Eliminar las direcciones IP cableadas del código.
- No almacenar direcciones IP.
- Las direcciones “IPv4-mapped IPv6 addresses” no siempre funcionan:
  - No siempre están implementadas.
  - En algunos nodos están deshabilitadas por seguridad.
- Al portar aplicaciones revisar exhaustivamente el código.