Adaptación de aplicaciones a IPv6

extensión de la interfaz de sockets

Eva M. Castro

ecastro@dit.upm.es

Contenido

* Introducción
  - Modelo de sockets BSD
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets
* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack
* Caso de estudio: ISABEL
* Conclusiones

Introducción

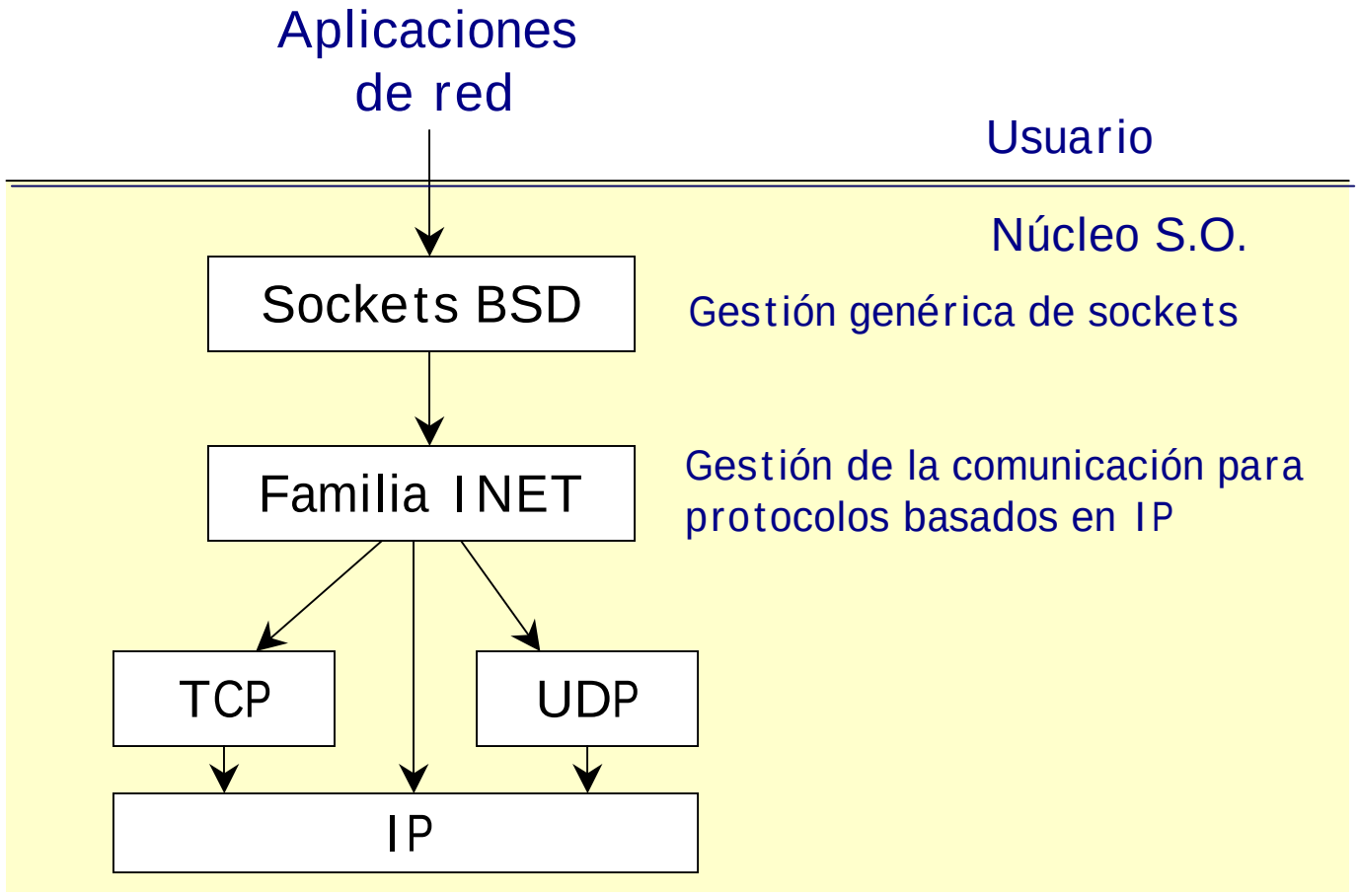
Aplicaciones IPv4 pueden trabajar en IPv6:

1. Adaptando la parte de red para que utilice direcciones IPv6.
   Utilizando la interfaz de sockets:
   * RFC 2553: Basic socket interface extensions for IPv6 (Draft Oct 2000).
   * RFC 2292: Advanced sockets API for IPv6 (Draft Nov 2000).

2. Reingeniería de la aplicación para que utilice las nuevas posibilidades, además del mayor espacio de direccionamiento:
   * QoS: Etiquetas de flujo y prioridades.
   * Movilidad.
   * Multihoming.
   * Comunicación anycast.
   * Seguridad: autenticación y cifrado.
   * ...

Modelo de sockets BSD

Abstracción de los detalles de la comunicación utilizando una interfaz común



Contenido

* Introducción
  - Modelo de sockets BSD

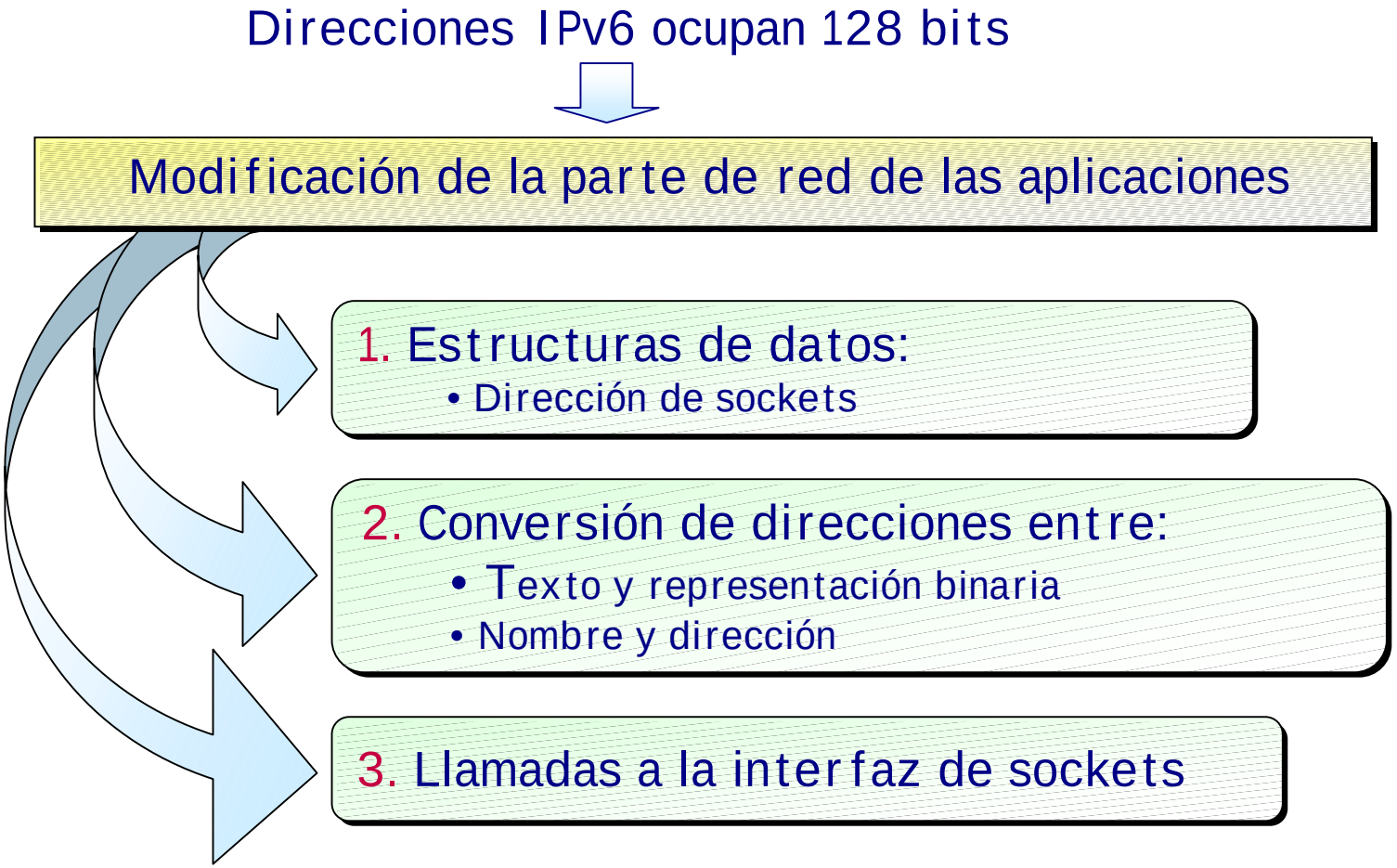
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets

* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack

* Caso de estudio: ISABEL

* Conclusiones

Adaptación de aplicaciones IPv4 a IPv6 (extensión a la interfaz de sockets)

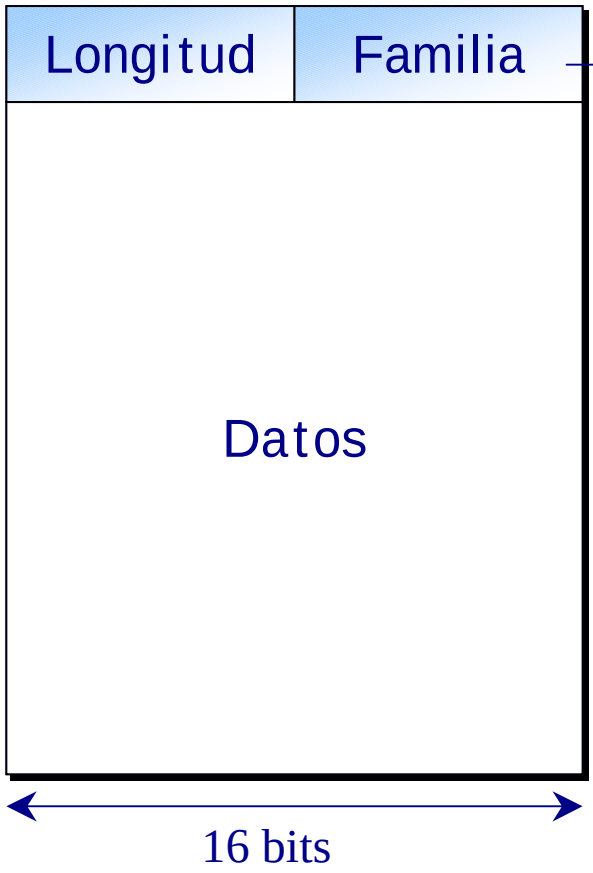


Contenido

* Introducción
  - Modelo de sockets BSD
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets
* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack
* Caso de estudio: ISABEL
* Conclusiones

Estructura de dirección genérica de socket

sockaddr




```
struct sockaddr {
   uint8_t       sa_len;
   sa_family_t   sa_family;
   char          sa_data[14];
};
```

Estructura de dirección de socket IPv4

sockaddr_in


```
struct sockaddr_in {
   uint8_t          sin_len;
   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

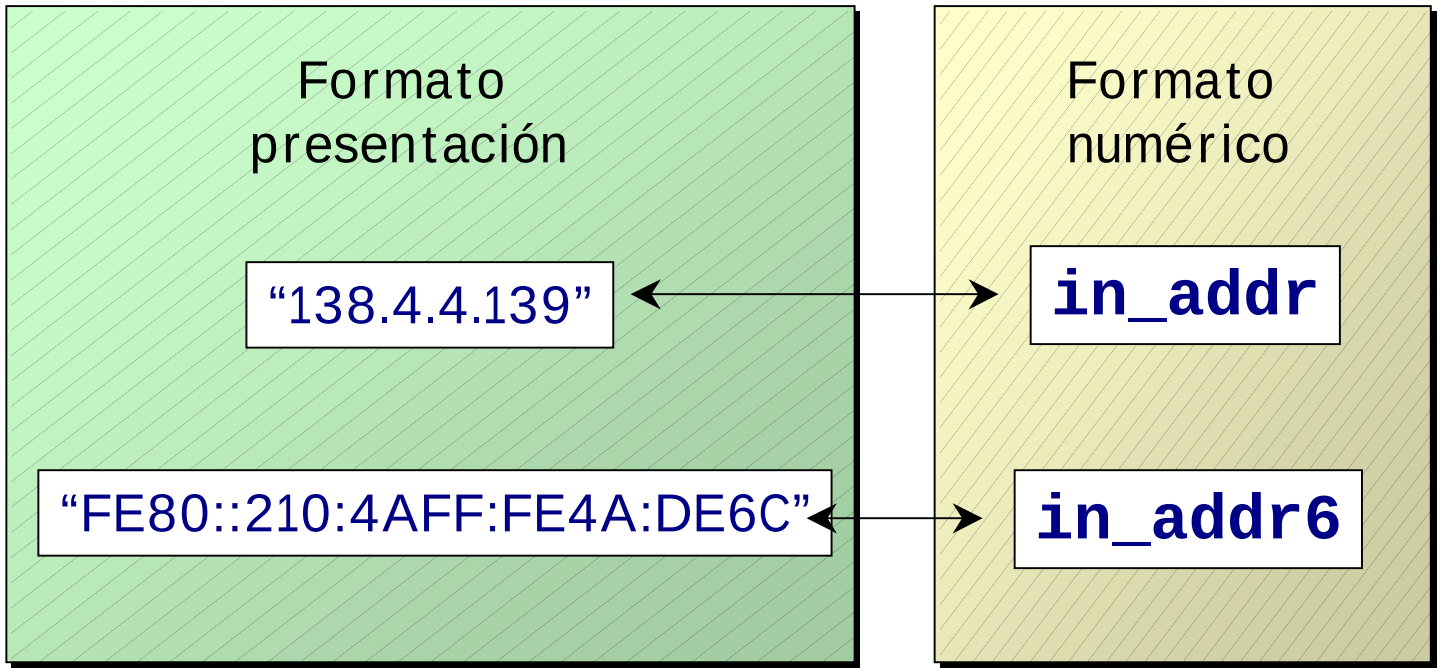

```
struct sockaddr_in6 {
   uint8_t           sin6_len;
   sa_family_t       sin6_family;
   in_port_t         sin6_port;
   uint32_t          sin6_flowinfo;
   struct in6_addr   sin6_addr;
};

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

Contenido

* Introducción
  - Modelo de sockets BSD
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets
* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack
* Caso de estudio: ISABEL
* Conclusiones

Conversión de direcciones: Texto y representación binaria



Representación binaria de la dirección IPv4 (32 bits)

Representación binaria de la dirección IPv6 (128 bits)

Conversión de direcciones: Texto y representación binaria

1. Texto a representación binaria **(in_addr, in6_addr)**

Sólo IPv4

```
int         inet_aton (const char *strptr,
                       struct in_addr *addrptr);
in_addr_t   inet_addr (const char *strptr); /*obsoleto*/
```

IPv4 & IPv6

```
int         inet_pton (int family, const char *strptr,
                       void *addrptr);
```

2. Representación binaria **(in_addr, in6_addr)** a texto

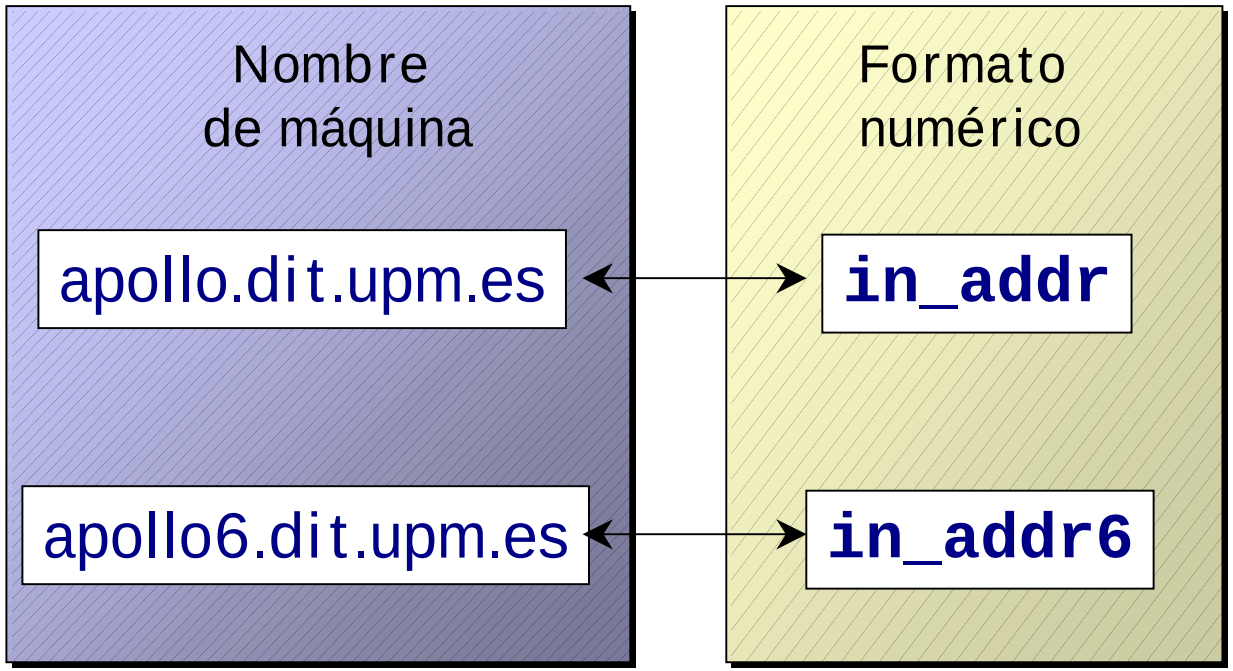
Sólo IPv4

```
int         inet_ntoa (struct in_addr inaddr);
```

IPv4 & IPv6

```
int         inet_ntop (int family, const char *strptr,
                       void *addrptr);
```

Conversión de direcciones: Nombre de máquina y dirección IP



Representación binaria de la dirección IPv4 (32 bits)

Representación binaria de la dirección IPv6 (128 bits)

Conversión de direcciones: Nombre de máquina a dirección IP

```
struct hostent *gethostbyname(const char *hostname);
```

* Consulta al DNS por una entrada de tipo A

"hostname.domain" ⟶ Dirección IPv4

* Consulta al DNS por una entrada de tipo AAAA, definir la opción **RES_USE_INET6**:

"hostname.domain" + **RES_USE_INET6** ⟶ Dirección IPv6 (si no hay, dirección IPv4-mapped IPv6)

Conversión de direcciones: Nombre de máquina a dirección IP

```
struct hostent *gethostbyname2(const char *hostname, int family);
```

* Consulta al DNS por una entrada de tipo A (**family=AF_INET**):

"hostname.domain" → Dirección IPv4

"hostname.domain" → **RES_USE_INET6** → Dirección IPv4-mapped IPv6

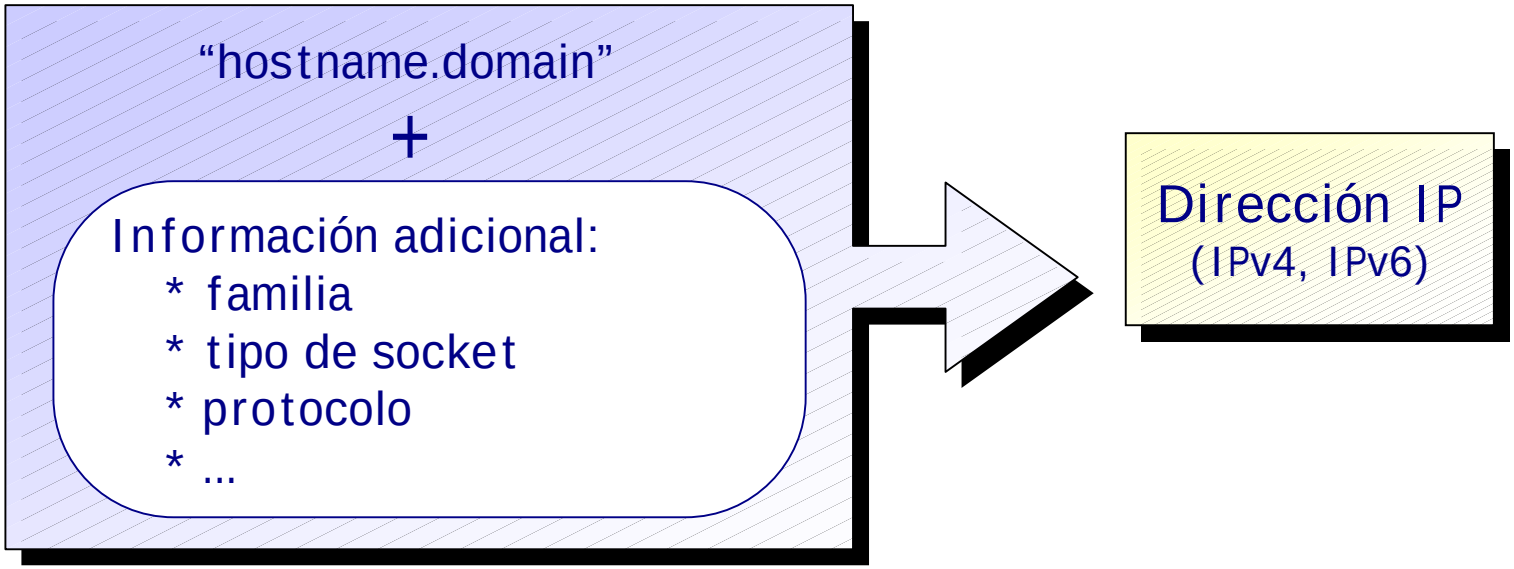
* Consulta al DNS por una entrada de tipo AAAA (**family=AF_INET6**):

"hostname.domain" → Dirección IPv6

Conversión de direcciones: Nombre de máquina a dirección IP

```
int getaddrinfo (const char *hostname, const char *service,
                 const struct addrinfo *hints,
                 const struct addrinfo **result);
```

* Función independiente del protocolo:



Conversión de direcciones: Dirección IP a nombre de máquina

```
struct hostent *gethostbyaddr(const char *addr,
                              size_t len,
                              int family);
```

* Dirección IPv4:

```
addr   => in_addr
len    => sizeof(in_addr)
family => AF_INET
```

* Dirección IPv6:

```
addr   => in6_addr
len    => sizeof(in6_addr)
family => AF_INET6
```

IPv4-mapped IPv6
Dirección compatible
} => Consulta en el dominio de direcciones IPv4

Conversión de direcciones: Dirección IP a nombre de máquina

```
int getnameinfo (const struct sockaddr *sockaddr,
                 socklen_t addrlen,
                 char *host, size_t hostlen,
                 char *serv, size_t servlen, int flags);
```

* Función independiente del protocolo

  - Dirección IPv4:

```
sockaddr => sockaddr_in
len => sizeof(sockaddr_in)
```

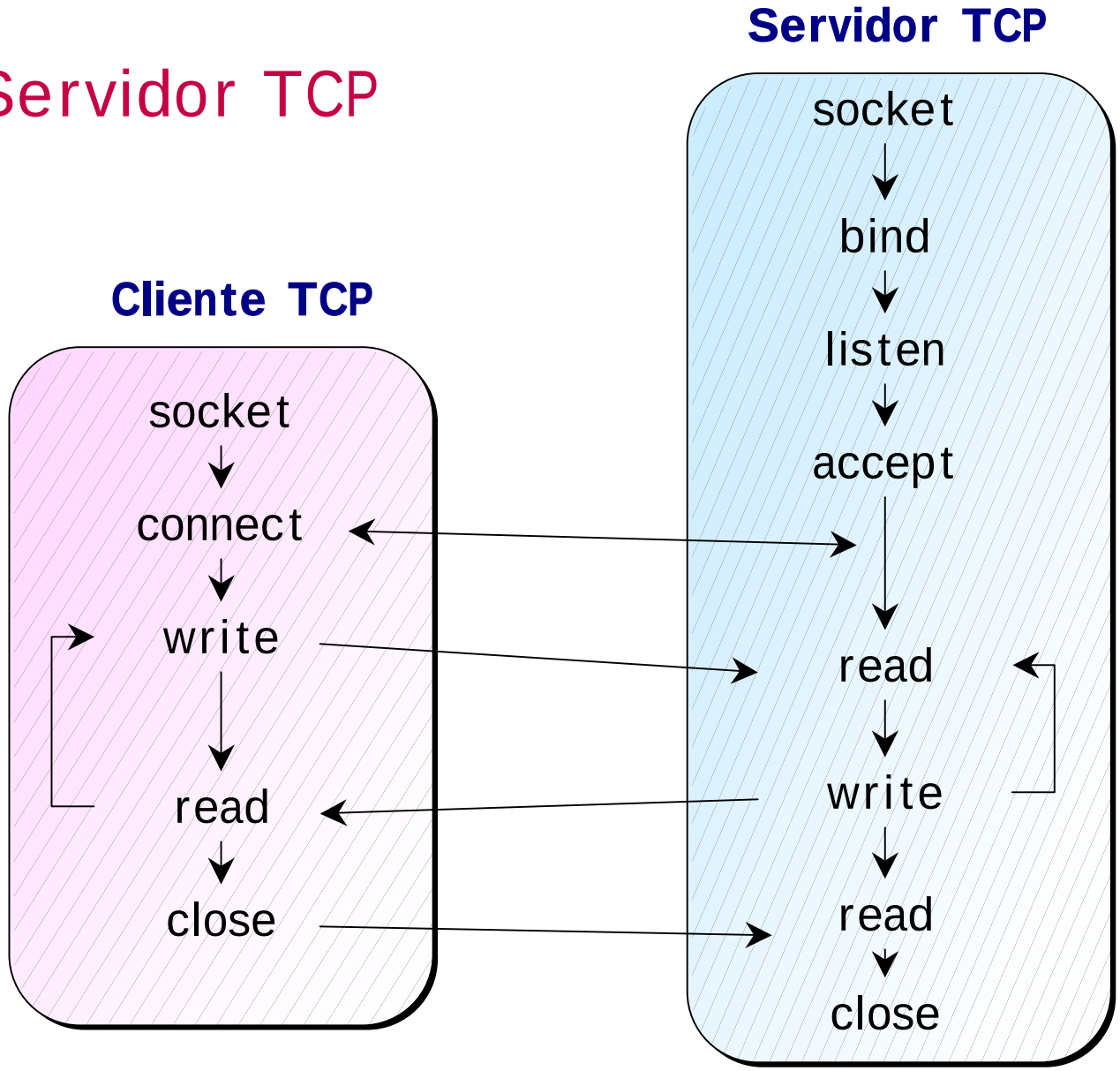
  - Dirección IPv6 :

```
sockaddr => sockaddr_in6
len => sizeof(sockaddr_in6)
```

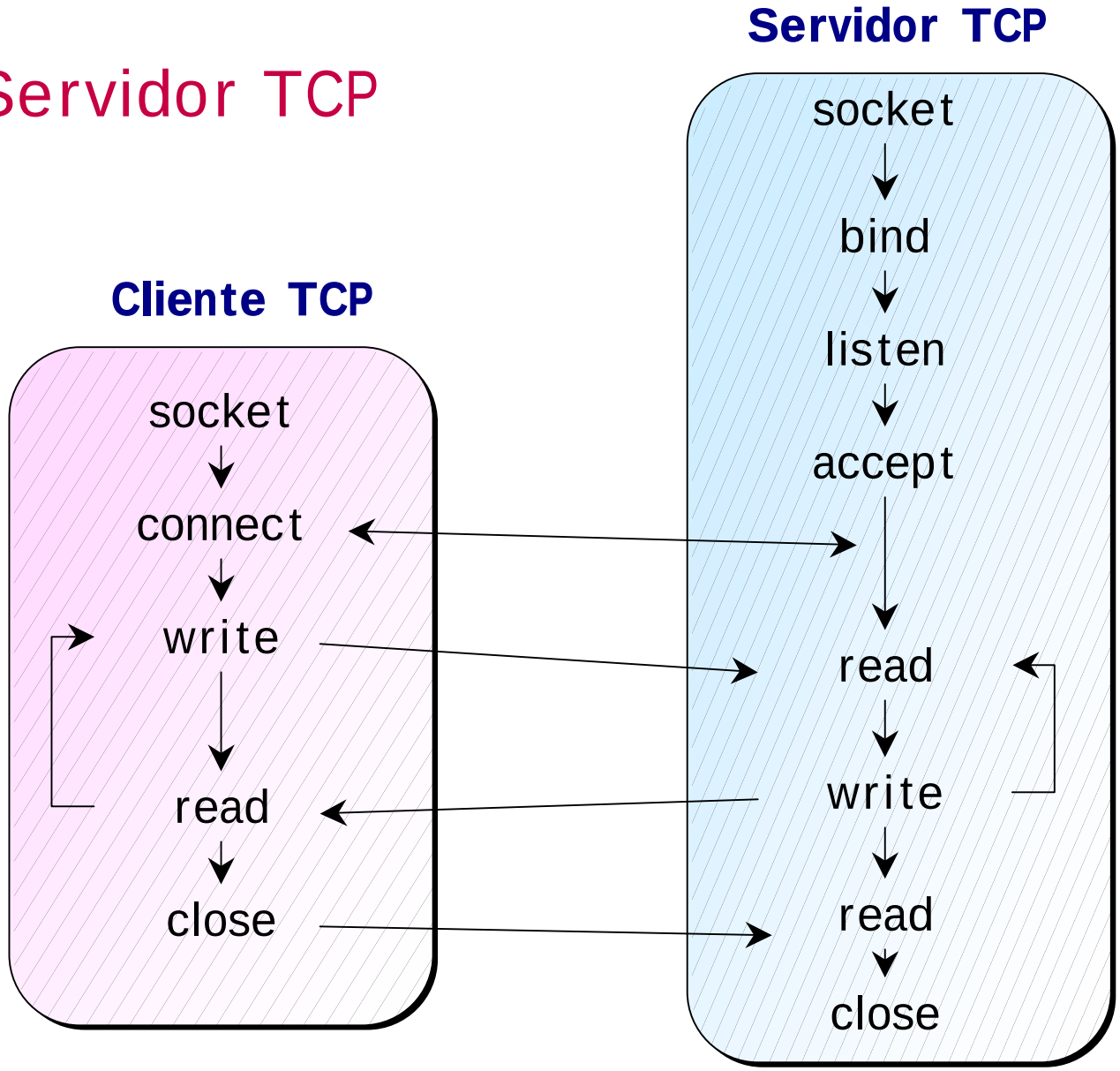
Contenido

* Introducción
  - Modelo de sockets BSD
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets
* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack
* Caso de estudio: ISABEL
* Conclusiones

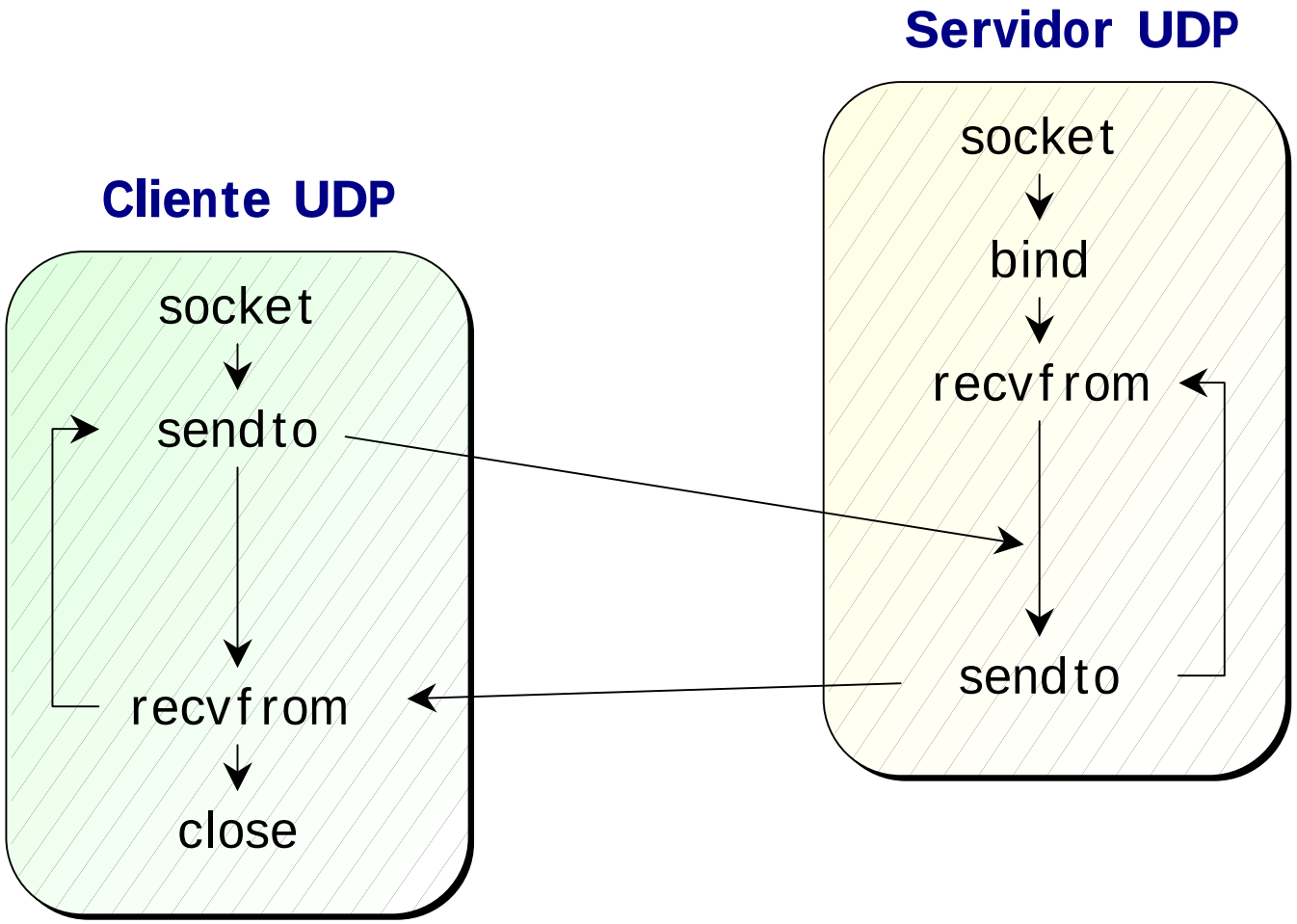
Cliente-Servidor TCP en IPv4



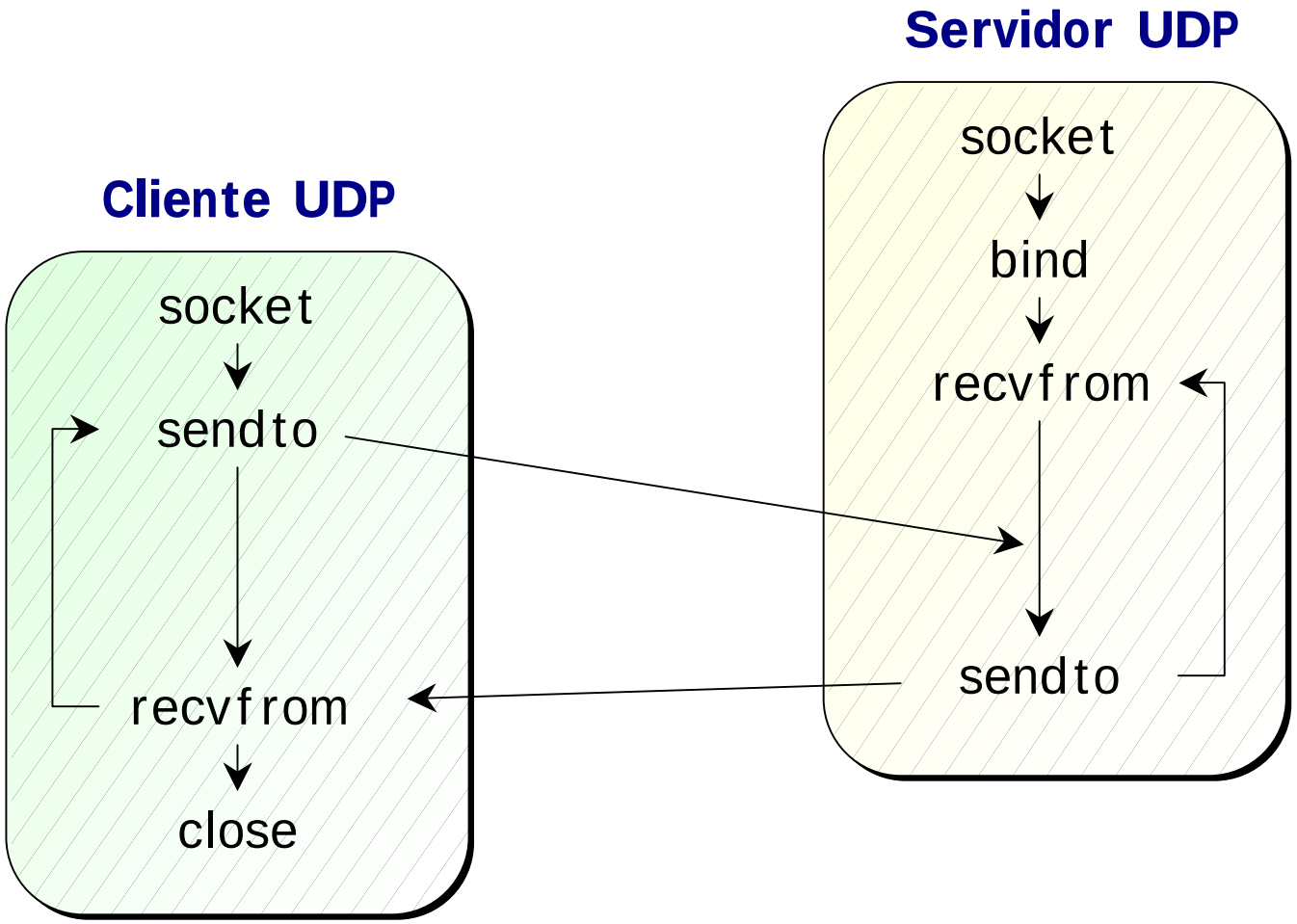
Cliente-Servidor TCP en IPv6



Cliente-Servidor UDP en IPv4



Cliente-Servidor UDP en IPv6



Mismas llamadas de socket en IPv4 e IPv6

Algunos valores de los argumentos del API varían:

* Familia de la dirección, tipo de socket y protocolo en la llamada **`socket`**:

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

family \ type	SOCK_STREAM	SOCK_DGRAM	SOCK_RAW
AF_INET	TCP	UDP	IPv4
AF_INET6	TCP	UDP	IPv6

protocol

* Puntero a la estructura **`sockaddr`** y su longitud:

IPv4:
- Casting `(sockaddr_in *)` a `(sockaddr *)`
- Longitud => `sizeof(sockaddr_in)`

IPv6:
- Casting `(sockaddr_in6 *)` a `(sockaddr *)`
- Longitud => `sizeof(sockaddr_in6)`

Contenido

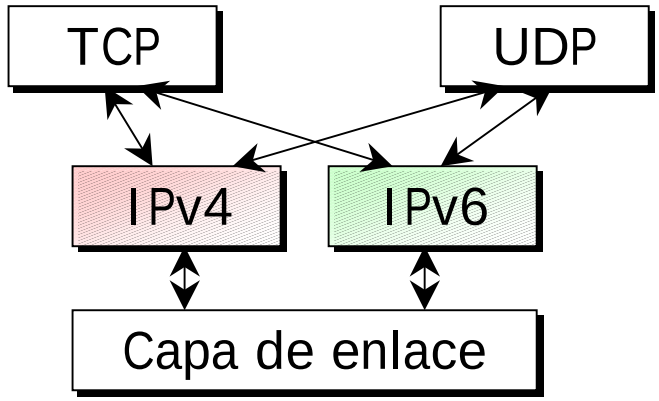
* Introducción
  - Modelo de sockets BSD
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets

* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack
* Caso de estudio: ISABEL
* Conclusiones

Interoperabilidad entre IPv4 e IPv6

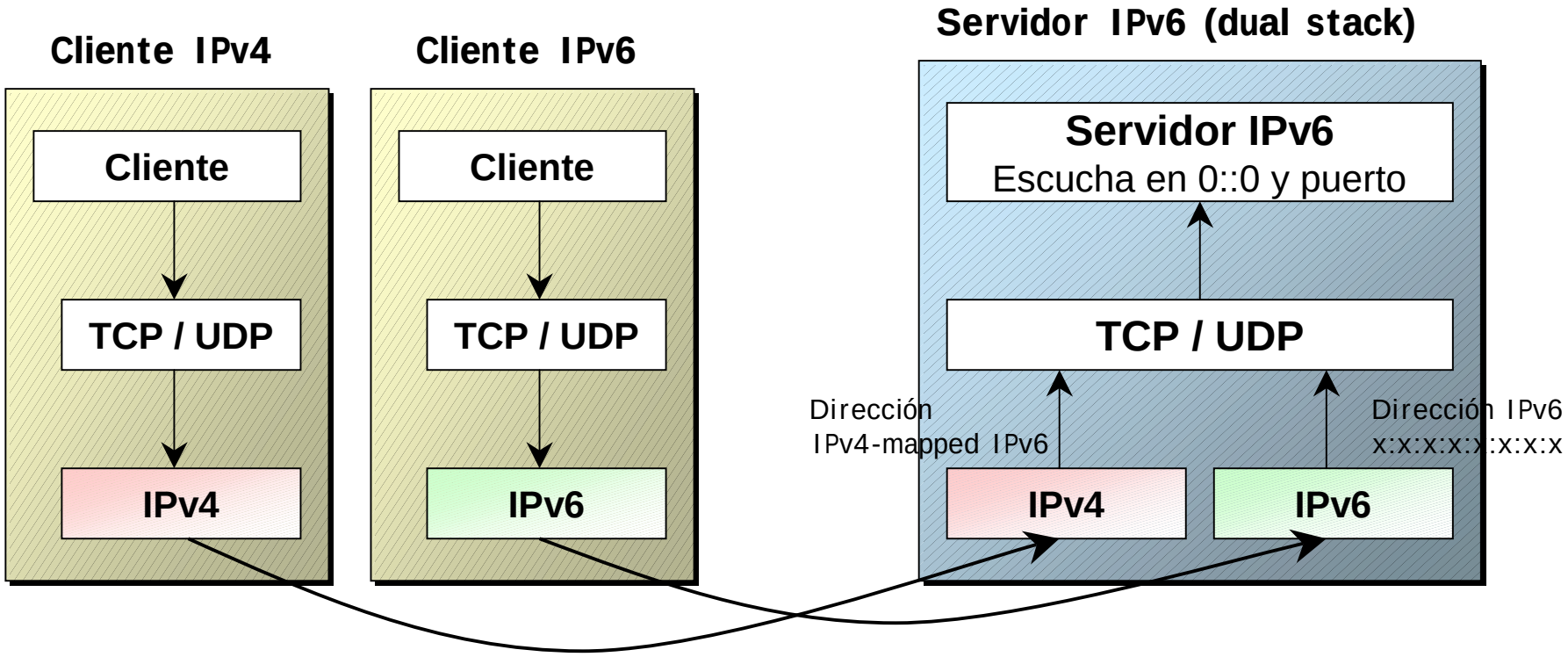
* Durante el período de transición las aplicaciones sobre IPv4 existentes deberían funcionar con nuevas aplicaciones sobre IPv6.

Ambos protocolos IPv4 & IPv6 (Dual stack)



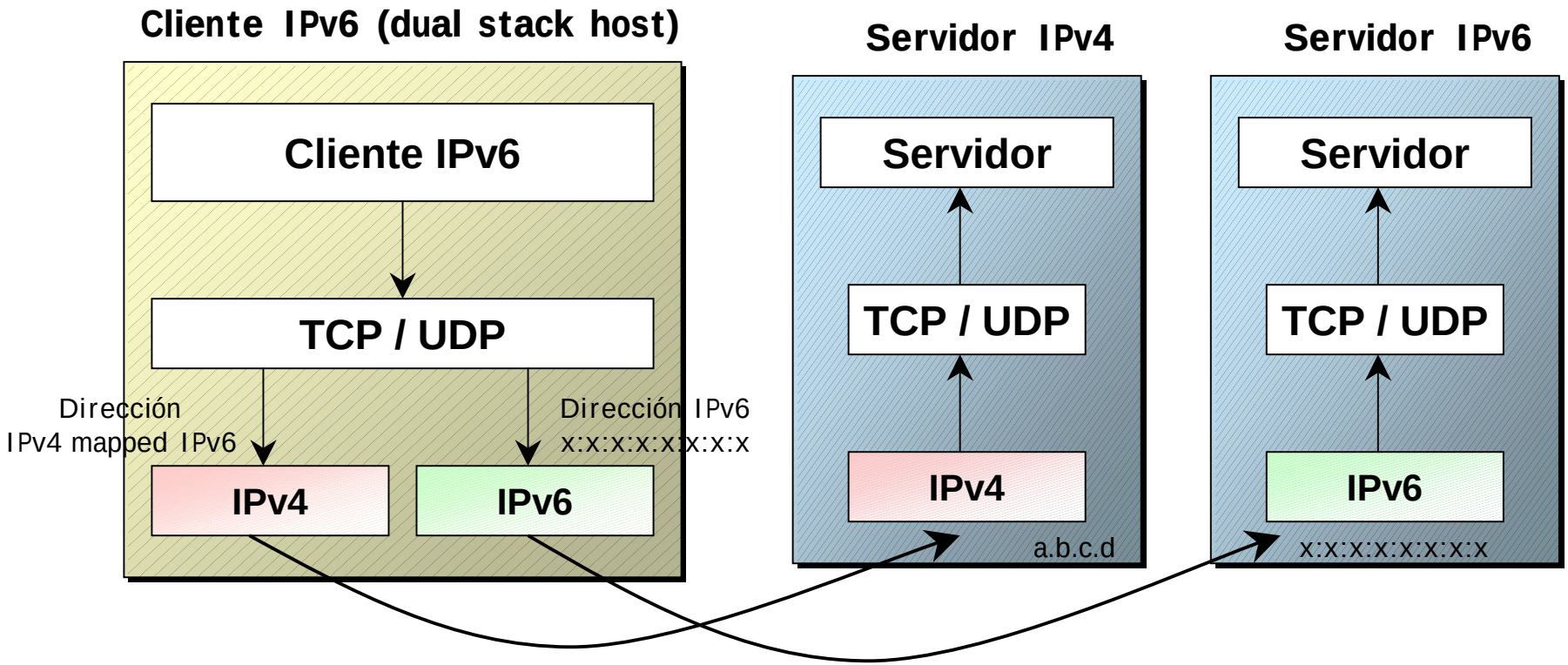
Cliente IPv4 – Servidor IPv6

* Servidores IPv6 sobre una máquina con dual stack aceptan conexiones de clientes IPv4 e IPv6.



Cliente IPv6 – Servidor IPv4

* Clientes IPv6 en una máquina con dual stack pueden conectarse a servidores IPv4 e IPv6.

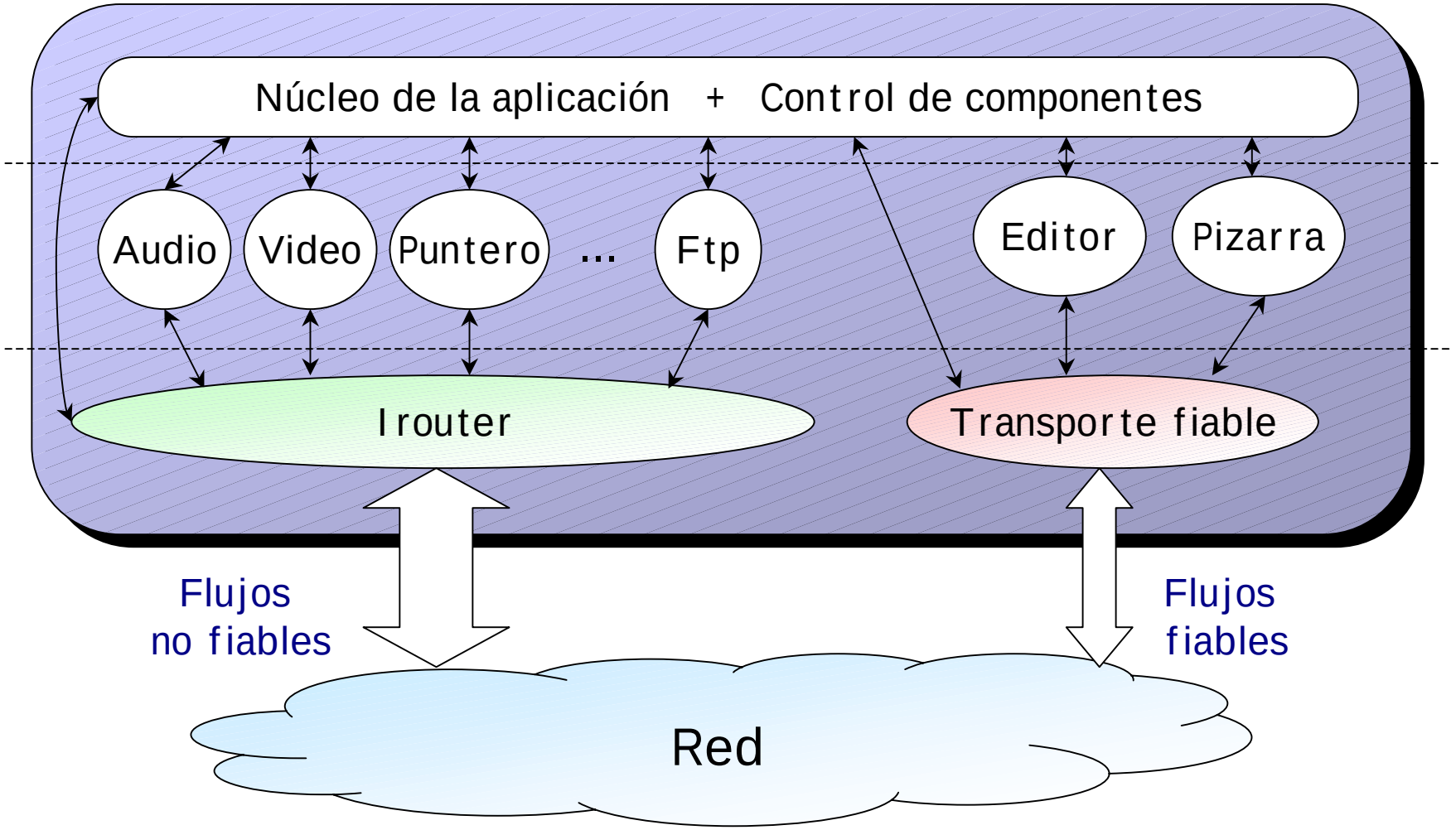


Contenido

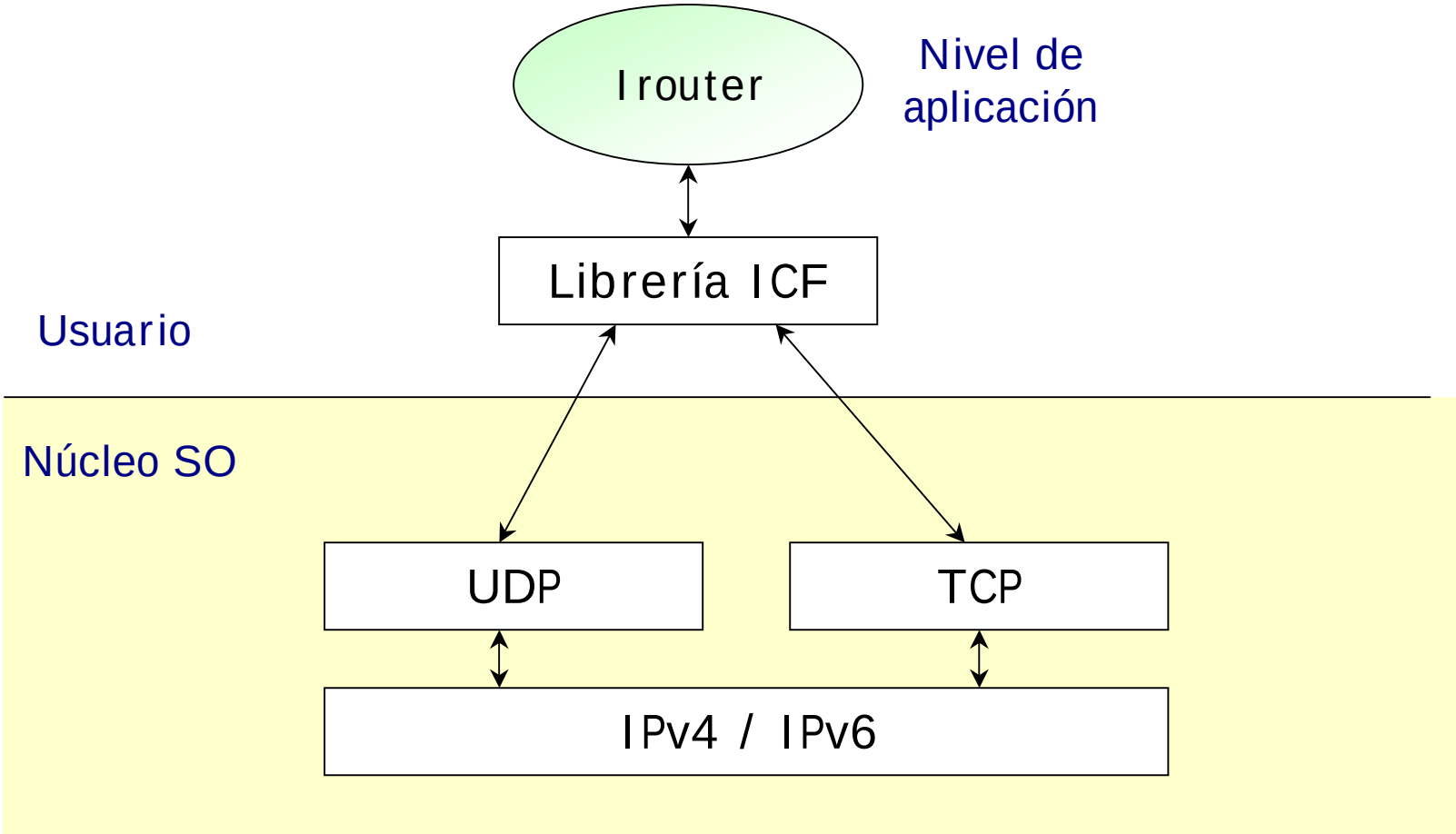
* Introducción
  - Modelo de sockets BSD
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets
* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack

* Caso de estudio: ISABEL
* Conclusiones

Caso de estudio: ISABEL



Caso de estudio: ISABEL



Contenido

* Introducción
  - Modelo de sockets BSD
* Adaptación de aplicaciones IPv4 a IPv6.
  Extensión de la interfaz de sockets:
  - Estructuras de datos
  - Conversión de direcciones
  - Llamadas a sockets
* Interoperabilidad entre IPv4 e IPv6
  - Dual-stack
* Caso de estudio: ISABEL
* Conclusiones

Conclusiones

* Portabilidad.

* Aplicaciones independientes del protocolo IPv4 o IPv6:
  Librería para el manejo de sockets, ocultando la dependencia del protocolo y simplificando el código de la aplicación.

* Utilización de las funciones **`getaddrinfo`**, **`getnameinfo`** en vez de **`gethostbyname`** y **`gethostbyaddr`**.