

TS : Sockets

Introducción

socket → punto final de comunicación
(direc IP + Puerto + protocolo)

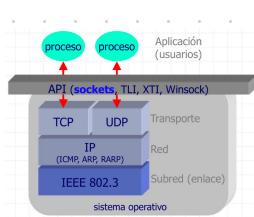
- El alcance de sockets es **límite de**
protocolo de PCI (ancho ≈ 25)
Lenguaje de Programación
distintivo

- Es una abstracción que
ofrece interfaces de acceso a los servicios
de la red en el nivel transporte

↳ Representa un extremo de una comunicación
bidireccional con una dirección asociada

Membrana entre ① y 2 de comunicación

- Sujeto a proceso de estandarización dentro de ROSA
Sockets ISO/IEC vs. Sockets BSD (Berkeley)
estandarización originales



- socket → ligado → dirección IP: Puerto
Protocolo (TCP, UDP, ...)
- 2^{16} puertos posibles
- No se puede reabrir un Puerto

↳ ya asignado a otro proceso

Modelo Cliente - Servidor



- ④ q ejec en nodo 1 req
+ proporciona desc. recurso o
contenimiento ESPERANDO peticiones

- 5. interactivo → Ser: recibe y atiende petición
6. concurrente → Ser: recibe petición
genera

⑦ encargado de atención



- ④ q ejec. en otro nodo (o el mismo)
y realiza peticiones al servidor

⑧ INICIA y TERMINA el diálogo

Implementación



- 1) creación + enlace
puerto de servicio de un socket

- 2) Descripción del servidor
(de su terminal de ejec. o
de su terminal de ejec.)

- 3) Bucle:
esperar petición → Tratadora
enviar respuesta → Cola + manejar respuesta



- 1) creación socket local
puerto de servicio de un socket

- 2) Averiguar direc. IP

- 3) env del P

- 4) Esperar resultado

- 5) EXPLORACIÓN del resultado

Conceptos Básicos sobre Sockets

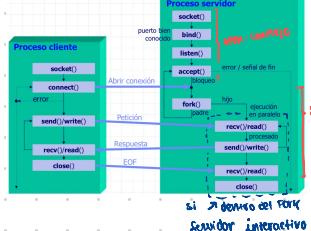
Dominios de comunicación

Tipos (estilos) de Sockets

Direcc. de Sockets

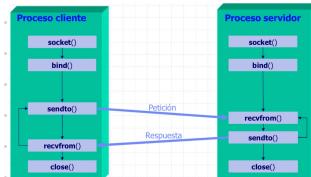
Escenario uso de Sockets stream

TCP



Escenario uso de Sockets datagramas

UDP



Dominios de Comunicación

Dominió
representa
fam. de protocolos
y codic. Dom → propio formato de direc.

servicio de sockets

Dom → donde crea/soc
solo se miden PCA sockets = Dom.

- Dominió PF_UNIX o PF_LOCAL → dentro de una máquina
- Dominió PF_INET → usando protocolos TCP/IP
- Dominió PF_NS → XEROX Network Source (NS) dominio
- Dominió PF_APPLETALK → protocolo Appletalk de Apple

Tipos e Estilos de Sockets

conjunto propiedades del servicio q se deben

Tipo

stream

datagram

raw

sequenced packet

reliable delivered message

Sockets stream (SOCK_STREAM):

- Representa un circuito orientado a conexiones
- Al conectar → se establece una conexión y se mantiene todo lo largo
- Propiedades
 - orientado a servicios → Full-duplex
 - Full → no se pierden/duplicitan datos
 - Secuencial → asegura orden entrega datos
 - NO mantiene separación entre sockets (solo stream)
 - Permite ACK para de bandas (out of band) → msg al margen
- Dom PF_INET → protocolo TCP y también PF_UNIX → STIFIO TUN DUPLEX

Sockets datagram (SOCK_DGRAM)

- red resuelve una red basada en datagramas (no orientada a conexiones)
- cada socket (P) tiene su propia dirección
- Propiedades
 - Sin conexión
 - NO Full → se pierden/duplican datos
 - NO se garantiza recepción secuencial
 - Mantiene separación entre mensajes
- Dom PF_INET → Protocolo UDP

Rollo (SOCK_RRDY)

- Permite el acceso a los protocolos de nivel más bajo (eth, internet, ...)
- requieren permisos root
- puede usarse para implementar servicios de transmisión o generando un enrutador

Otros tipos

- Sequenced packets (SOCK_SEQPACKET)
 - PF_NS
 - socket stream con un entre P.
 - Reliable delivery Message (SOCK_RDM)
 - Transferir. D generando su enrutador
 - Nivel de red implementado

Gestión de direc. de Sockets

T5: Sockets

Mapa de la API de Sockets

Aceptar una conexión

`accept()`: Servidor sockets STREAM tras conexión.

```
import socket
s = socket.socket()
s.bind('...')

Desarrolla una pareja de sockets:
• El nuevo socket que se crea al aceptar la conexión
• La dirección remota de dicha conexión (dirección)
• En caso de error lanza una excepción socket.error:
    • EBADF - s no es un descriptor de fichero válido
    • ENOTSOCK - s no es un socket
    • EPROTONOSUPPORT - el socket no soporta esta operación
    • ENOPROTOOPT - s no tiene opción y no hay conexiones pendientes
```

Semántica de accept():

- Solo se usa para sockets STREAM
- Es la 1ª ejecución de la familia creada (usarse)

- Cuando se produce la conexión → sev obtiene
 - descrip. socket remoto cliente
 - Nueva descripción → conectado al socket cliente
- Cuando devuelve accept() → quedan activos 2 sockets. Pueden ser sockets originales → OBTENER NUEVAS CONEXIONES
- Sockets Nuevo → GET/leer datos

Gracias a esto 3 servidores simultáneos tienen sockets concurrentes

Cierre de un socket

`close()` e `shutdown()`

```
import socket
s.close() → más de gente

import socket
s.shutdown(socket.SHUT_RDWR) → para orientados a conexión es importante hacer el "cierre educado"
    • El parámetro como dice cómo cierra:
        - si se establece en 0: se cierra el socket y se rechazan las conexiones
        - si se establece en 1: se cierra el socket y se dejan las conexiones
        - si se establece en 2: se cierra el socket y se dejan las conexiones por transmitir y reconocimientos o retransmisiones de datos
        - SHUT_RD → SHUT_RD + SHUT_WR es decir, solo a cerrar la conexión con close()
```

```
import socket
s.settimeout(5) → fija el límite de un intento de conexión (en segundos).
```

Envío y recepción de datos

Envío:

- `send(data[, flags[, sendto]])`
 - = que se usa en los clientes OS-TCP/IP USA DESCRIPCIÓN DE LOS SOCKETS
- Recepción:
 - `recv(nbytes[, flags])`, `recvfrom()`
 - = que se usa en los clientes OS-TCP/IP USA DESCRIPCIÓN DE LOS SOCKETS

Sockets STREAM:

Envío:

- ```
import socket
s.send(data[, flags])
 • Devuelve un int de bytes enviados
 • En caso de error lanza una excepción socket.error y fija el valor adecuado
 • Con flags=0, send() es equivalente a send() en objetos de tipo fichero (los devueltos por open())
```

##### Recepción:

- ```
import socket
s.recv(nbytes[, flags])
    • Con flags=0, recv() es equivalente a read() en objetos de tipo fichero (los devueltos por open())
```

Sockets DATAGRAMA:

- No se establece conexión (`connect()`) / `accept()`
 - se usar un socket para transferir hasta con crear el socket y reservar la dirección (bind())
- Envío:


```
import socket
s.sendto(data[, destination[, flags]])
    • En dirección la dirección del destinatario.
```
- Recepción:


```
import socket
s.recvfrom(nbytes[, flags])
    • Si devuelven la dirección del destinatario, igual que sendto()
    • Puede usarse recv() si no importa la dirección del destinatario, e incluso read() si no se usan los flags.
```

Sockets como ficheros

• Se puede configurar timeout conexión

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
f = s.makefile('rwb', 0)
    • Desarrolla un objeto fichero
```

• Pueden permisos: llaves de seguridad de ficheros: open()

Sockets en lugar de Pipes

2 sockets → estructura > pipe

```
import socket
s1, s2 = socket.socketpair(socket.AF_UNIX, socket.SOCK_STREAM)
    • Desarrolla una pareja de sockets (stax2) conectados entre si, igual que en pipes
    • En caso de error lanza una excepción socket.error y fija el valor adecuado:
        • EMFILE - el proceso tiene demasiados descriptores abiertos
        • ENAMOUNT - dominio no soportado
        • EPROTOSUPP - protocolo no soportado
        • EPROTOSUPP - el protocolo no permite crear parejas de sockets
```

- crea una pareja de sockets sin nombre conectados
- diferencia: sockets son BI-DIRECC.

Direcc. Local y remota de un socket

```
import socket
direccion_local = s.getsockname()
direccion_remota = s.getpeername()
    • Devuelve una dirección IPv4 ó IPv6
    • En caso de error lanza una excepción socket.error y fija el valor adecuado:
        • EBADF - s no es un descriptor de fichero válido
        • ENOTSOCK - s no es un socket
        • EPROTOSUPP - el socket no soporta direcciones
        • ENOSRBUFS - no hay suficientes buffers de entrada/salida para satisfacer la petición
```

Comandos y llamadas de utilidad

ver conexiones abiertas del sistema → netstat -a
ver llamadas al sistema resulta un `(2) stat -l` → sincronizar un descriptor de fichero con `sync (fd)`

Config. de opciones de sockets

3 niveles de opciones dependiendo del protocolo

```
opciones independientes del protocolo: socket.BLOCKING
nivel de protocolo TCP: socket.TCP_NODELAY
nivel de protocolo IP: socket.IPPROTO_IP
socket.IPPROTO_TCP
socket.IPPROTO_IPV6
```

- consultar opciones asociadas a un socket: `socket.getsockopt()`
- modificar opciones asociadas a un socket: `socket.setsockopt()`

Ejemplos (nivel socket.BLOCKING):

- Para reutilizar direcciones: `socket.SO_REUSEADDR`

Gestión de Direcciones

- Direccionamiento en sockets
- Direccionamiento físico: convenciones decimal-punto → binario
- orden bytes
- Direccionamiento lógico: librería resolver
- Otras funciones
 - nombre host local
 - Direcc. local & remota socket

Direccionamiento en socket

USUALES:

- Formato decimal-punto: 172.24.9.200
- Formato dominio-punto: servidor.ceu.es

Necesidad conversión

Direccionamiento físico

- decimal-punto a binario: `inet_nton()`
- binario a dominio-punto: `inet_ntoa()`

```
import socket
domicinaria = socket.inet_nton(familia, dirección)
import socket
domicilio = socket.inet_ntoa(familia, dirección)
    • En caso de error lanza una excepción socket.error y fija el valor adecuado:
        • ENAMOUNT - dominio no soportado
        • ENOSRBUFS - el parámetro de dirección no tiene el formato adecuado
```

Familia de dirección. Sólo:

- AF_INET
- AF_INET6

Orden de los Bytes

Bytes o little Endian → BigEndian o Big Endian

```
import socket
socket.inet_aton('192.168.1.1') → socket.inet_aton('1.1.1.192')
socket.inet_ntoa('1.1.1.192') → socket.inet_ntoa('192.168.1.1')
```

little endian → bytes o little Endian

BigEndian → socket.inet_aton('192.168.1.1')

little endian → socket.inet_ntoa('1.1.1.192')

big endian → socket.inet_ntoa('192.168.1.1')

bytes → bytes o little Endian

T5: Societs

Gestión de Direcciones

4. Direccionamiento lógico: librería resolver

- Realiza las conversiones entre los sistemas mediante varios sistemas
 - ↳ FICHERO : OMS , AES/AES*
 - Llamados operan con  dominio-punto
dominio-punto
 - Resolver la marea Puerto es
Buenos Aires

DOMINIO-PUNTO a binario:

```
import socket
dirección_ip4 = socket.gethostbyname(nombre)
dirección_ip4, lista_aliases, lista_direccion_ip = socket.gethostbyname_ex(nombre)
```

• En caso de error lanzan una excepción `socket.error`. Si llaman a:

- Valores:
 - NO_RESOLVE_HOST
 - NO_DNS_CACHE_POUND
 - NO_DATA_TRY AGAIN
 - NO_RECOVERY

• Lista de direcciones IP del host
diferentes entre una sola

• Uso de nombres alternativos (alias)
generalmente vieja

• Nuevos cambios del host

→ extendido

• Servicios y Ayudas

Nombrar = a → fuentes

```
graph TD; Client[Client] -- "SYN, Port 12345" --> Server[Servidor]; Server -- "SYN-ACK, Port 12345" --> Client; Client -- "ACK, Port 12345" --> Server;
```

5. Otras Funciones

Número de host local

a máquina tal y como
sistema operativo

- El nombre de hosts bien config. sueltos coincide con el número mencionado en el DNS.

```
import socket  
nombre_canonico = socket.getfqdn(nombre="")
```

Info. de Dirección:

Posibles Valores de errno

errno	perrror()	Posible causa
EACCES	<i>permission denied</i>	El programa no tiene acceso a este socket.
EADDRINUSE	<i>socket address is already in use</i>	La dirección dada ya está siendo usada.
EADDRNOTAVAIL	<i>socket address not usable</i>	La dirección dada no está disponible en la máquina local.
EINVAL	<i>unsupported socket addressing family</i>	La familia de direcciones no está soportada o no es consistente con el tipo de socket dado.
EINVAL	<i>previous connection not yet completed</i>	El socket que se intenta cerrar ya no existe.
EBAUDF	<i>file or socket not open or user-level file</i>	El descriptor de fichero dado no se corresponde con un descriptor de (fichero) abierto.
ECONNABORTED	<i>connection aborted by local network software</i>	El sistema local de comunicaciones ha abortado la conexión.
ECONNREFUSED	<i>destination host refused socket connection</i>	La máquina destino ha rechazado la conexión del socket.
ECONNRESET	<i>connection reset by peer</i>	El socket remoto ha reiniciado la conexión.
ESTDADREQ	<i>socket operation requires destination address</i>	El socket necesita que se le suministre una dirección de destino.
EHOSTDOWN	<i>destination host is down</i>	El host de destino está apagado.
EHOSTUNREACH	<i>destination host is unreachable</i>	El host de destino es inalcanciable.

errno	perrror	Possible causa
EINPROGRESS	socket connection in progress	La conexión se ha iniciado y se está ejecutando. Puede que la llamada no haya llegado al puerto que la conexión. La conexión estará completa cuando un comando select() indique que el descriptor de fichero esté lista para lectura.
EISCONN	socket is already connected	El proceso llama a conectar() en un socket ya conectado.
EMSGSIZE	message too large for datagram socket	El mensaje es demasiado largo para accomodarlo en un socket de tipo datagrama.
ENETDOWN	local host's network down or inaccessible	El programa no puede hablar con el software de red en la máquina local, o bien la red local está apagada.
ENETRESET	remote host dropped network communications	El host remoto no se está comunicando por la red en este momento.
ENETREACH	destination network is unreachable	El host local no puede encontrar una ruta a la red de destino.
ENOBUFS	insufficient buffers in network software	El sistema operativo no tiene suficiente memoria para realizar la operación pedida.
ENPROTOPT	option not supported for protocol type	La opción (o el nivel de la misma) pedida sobre el socket no es válida.
ENOTCONN	socket not connected	El socket no está conectado.
ENOTSOCK	the descriptor not associated with a socket	El descriptor de fichero no corresponde a un socket sin un fichero, o bien el sin asignar.

errno	errorr()	Possible causa
EOPNOTSUPP	<i>operation not supported on socket</i>	La operación pedida no está soportada en el tipo de socket dado.
EPFNOSUPPORT	<i>unsupported socket protocol family</i>	La familia de protocolos dada es desconocida o bien el software de TCP/IP no la soporta.
EPPIPE	<i>broken pipe or socket connection</i>	El proceso remoto ha cerrado el socket (o la tubería) antes de que el proceso local haya terminado de anidar.
EPROTOSUPPORT	<i>unsupported socket protocol</i>	El protocolo dado es desconocido o bien el software de TCP/IP no lo soporta.
EPROTOTYPE	<i>protocol inconsistent with socket type</i>	Al llamar a socket() el protocolo no era ni 0 ni un valor consistente con el tipo de socket pedido.
ESHUTDOWN	<i>connection has been shut down</i>	La conexión ya ha sido cerrada.
ESOCKTNOSUPPORT	<i>socket type not allowed</i>	El proceso local ha pedido un tipo de socket no soportado por el software de TCP/IP.
ESYS	<i>operating system interface failure</i>	El software de TCP/IP, o el sistema operativo en el que se apoya, ha desenvuelto una condición anormal de falla. Considerar contactar con el fabricante.
ETIMEDOUT	<i>socket connection attempt timed out</i>	El socket destino no ha contestado a una petición de conexión.
EWOULDBLOCK	<i>socket operation would block</i>	El socket tiene activada la opción de entrada/salida bloquante, y esta llamada debería haberla.
		Este código no es un error.