Redes de computadoras

Semestre: 2018-2

SOCKETS

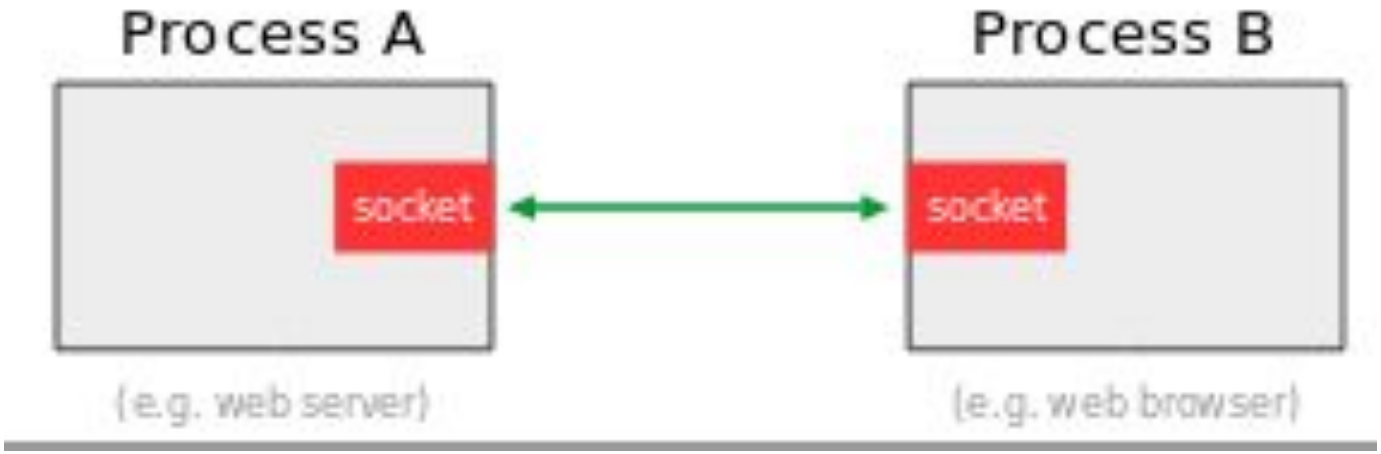
Sockets

Términos relacionados

- PILA TCP/IP
  - Capa de aplicación
    - API de Sockets
    - Espacio de usuario (Sistemas Operativos)
  - Capa de transporte
    - Llamadas al sistema(Espacio de Kernel)
    - TCP
    - UDP
  - Capa de red
    - IPv4
    - IPv6
  - ...

Definiciones de Socket

- Es una abstracción de una comunicación bidireccional(en ocasiones llamada asociación) entre dos procesos (remotos(?)).
- Es el punto final de una comunicación.
- Son una “Application Programming Interface (API)” para comunicación interprocesos (IPC)



Historia

A nivel de sistema operativo, podemos definir a los sockets como una de las muchas formas de comunicación interprocesos existentes.

Fue introducido originalmente en BSD 4.2 con la finalidad de crear un mecanismo IPC genérico.

A pesar de no ser un estándar, se ha convertido en un mecanismo de facto en la industria,

Familias de protocolos soportados

- PF_INET: IPv4 (o direcciones de 32 bits)
- PF_INET6: IPv6 (o direcciones de 64 bits)
- PF_UNIX: Mecanismo IPC en un mismo equipo
- PF_APPLETALK: Redes Appletalk
- PF_IPX: Redes Novell Netware
- PF_ALG: API criptografica del Kernel Linux
- Entre otros….

Tipos de Sockets

- SOCK_STREAM: Orientado a conexion (TCP)
- SOCK_DGRAM: Comunicación basada en datagramas(UDP)
- SOCK_RAW: Acceso directo a la capa de red (Por ejemplo, mensajes ICMP o paquetes IP adaptados)
- SOCK_PACKET: Acceso directo a la capa de enlace
- Entre otros...

Visualizar Sockets activos en el sistema

Los comandos netstat(también en Windows), ss y lsof, permiten visualizar sockets y su estado.

```
Netid  State      Recv-Q Send-Q Local Address:Port                 Peer Address:Port
u_str  ESTAB      0      0       * 741552                  * 742558
u_str  ESTAB      0      0      /run/systemd/journal/stdout 35221               * 38352
u_str  ESTAB      0      0       * 32788                   * 32789
u_str  ESTAB      0      0      @/com/ubuntu/upstart-session/1000/2256 31591                * 31590
u_str  ESTAB      0      0       * 742622                  * 742621
u_str  ESTAB      0      0       * 93251                   * 94214
u_str  ESTAB      0      0       * 35701                   * 35702
u_str  ESTAB      0      0      @/tmp/.X11-unix/X0 32750                 * 33802
u_str  ESTAB      0      0       * 42290                   * 42291
u_str  ESTAB      0      0       * 32779                   * 31895
```

API: Sockets de Berkeley/ POSIX

- Permite:
  - Configuracion de direcciones
  - Creadion de sockets, “Binding”, escucha
  - Inicializacion y aceptacion de una conexion
  - Envio y recepcion de informacion
  - Destruccion de sockets
  - Tecnicas de programacion (monitoreo, tratamiento de errores)

Empleando la API (En C)

int socket (int dominio, int tipo, int protocolo)

- dominio: Especifica la familia de protocolos(Por ejemplo PF_INET)
- tipo: Especifica el tipo de socket(SOCK_DGRAM, SOCK_STREAM)
- protocolo: Especifica el protocolo a ser usado por el socket (0 para default)

El valor de retorno de la funcion es un descriptor de archivos.

Implementación

SOCKET TCP

```
int sockDesc = socket(PF_INET, SOCK_STREAM, 0);
```

Estructuras de la API

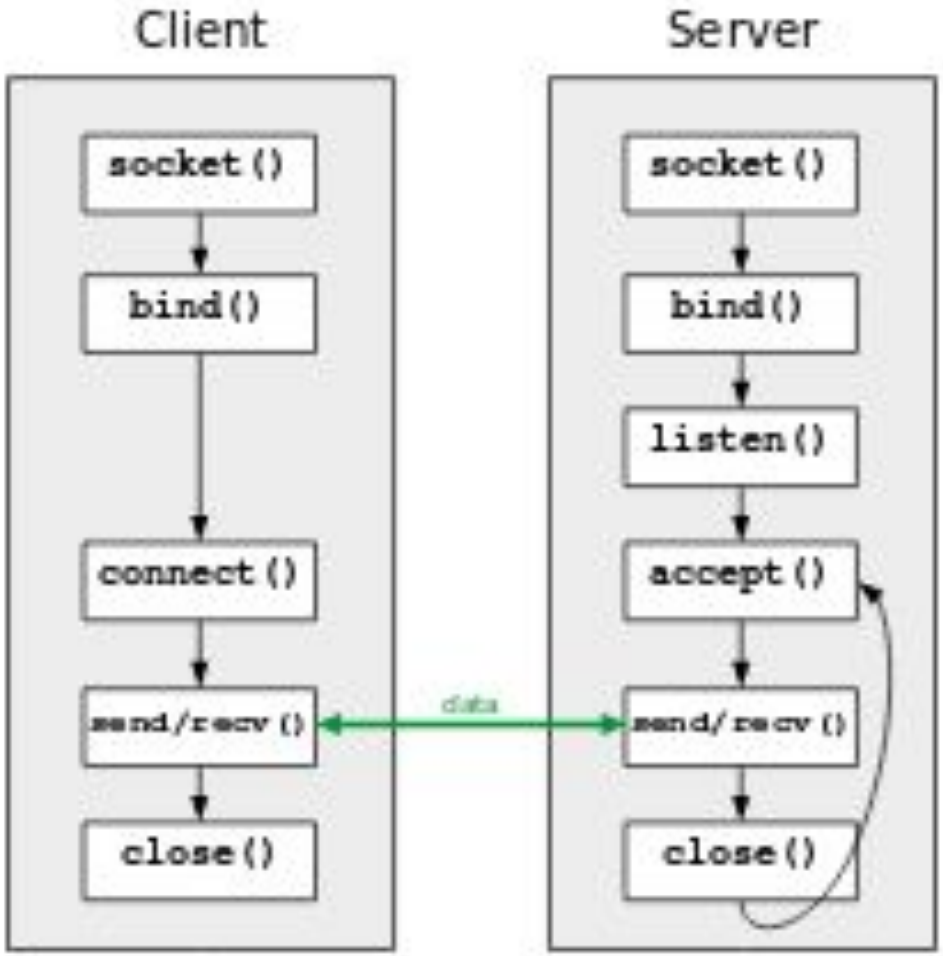
```
struct sockaddr_in locAddr;
locAddr.sin_family = AF_INET;
locAddr.sin_port = htons(0);
locAddr.sin_addr.s_addr = htonl(INADDR_ANY);
```

Implementación(cont.)

BINDING (Asociar una IP y un numero de puerto a un socket)

```
bind(sockDesc, (struct sockaddr*) &locAddr, sizeof(locAddr));
```

Funciones base



Ejemplo: Servidor ECHO (TCP)

```
int servSock = socket(PF_INET, SOCK_STREAM, 0);

struct sockaddr_in servAddr;
servAddr.sin_family = AF_INET;
servAddr.sin_port = htons(8080);
servAddr.sin_addr.s_addr = htonl(INADDR_ANY);

bind(servSock, (struct sockaddr*) &servAddr,
     sizeof(servAddr));
```

Ejemplo: Servidor ECHO (TCP)

```
listen(servSock, MAXPENDING);

for(;;) {
   struct sockaddr_in cltAddr;
   int cltAddrLen = sizeof(cltAddr);
   int cltSock;

   cltSock = accept(servSock,
                    (struct sockaddr*) &cltAddr,
                    &cltAddrLen);
```

Ejemplo Cliente

```
int localSock = socket(PF_INET, SOCK_STREAM, 0);
```

```
struct sockaddr_in localAddr;
localAddr.sin_family = AF_INET;
localAddr.sin_port = htons(0);
localAddr.sin_addr.s_addr = htonl(INADDR_ANY);

bind(localSock, (struct sockaddr*) &localAddr,
     sizeof(localAddr));
```

Ejemplo Cliente

```
struct sockaddr_in srvAddr;
srvAddr.sin_family = AF_INET;
srvAddr.sin_port = htons(8080);
srvAddr.sin_addr.s_addr = inet_addr("1.2.3.4");

connect(localSock, (struct sockaddr*) &srvAddr,
    sizeof(srvAddr));


        char msg[] = "Ping-Pong Message!";
        send(localSock, msg, strlen(msg), 0);
```

Respondiendo...

```
#define   BUFSIZE   32
char buf[BUFSIZE];
int bytesRecv;

bytesRecv = recv(cltSock, buf, BUFSIZE, 0);
while(bytesRecv > 0) {   // 0 means end of transm.
  send(cltSock, buf, bytesRecv, 0); // ECHO
  bytesRecv = recv(cltSock, buf, BUFSIZE, 0);
}
```