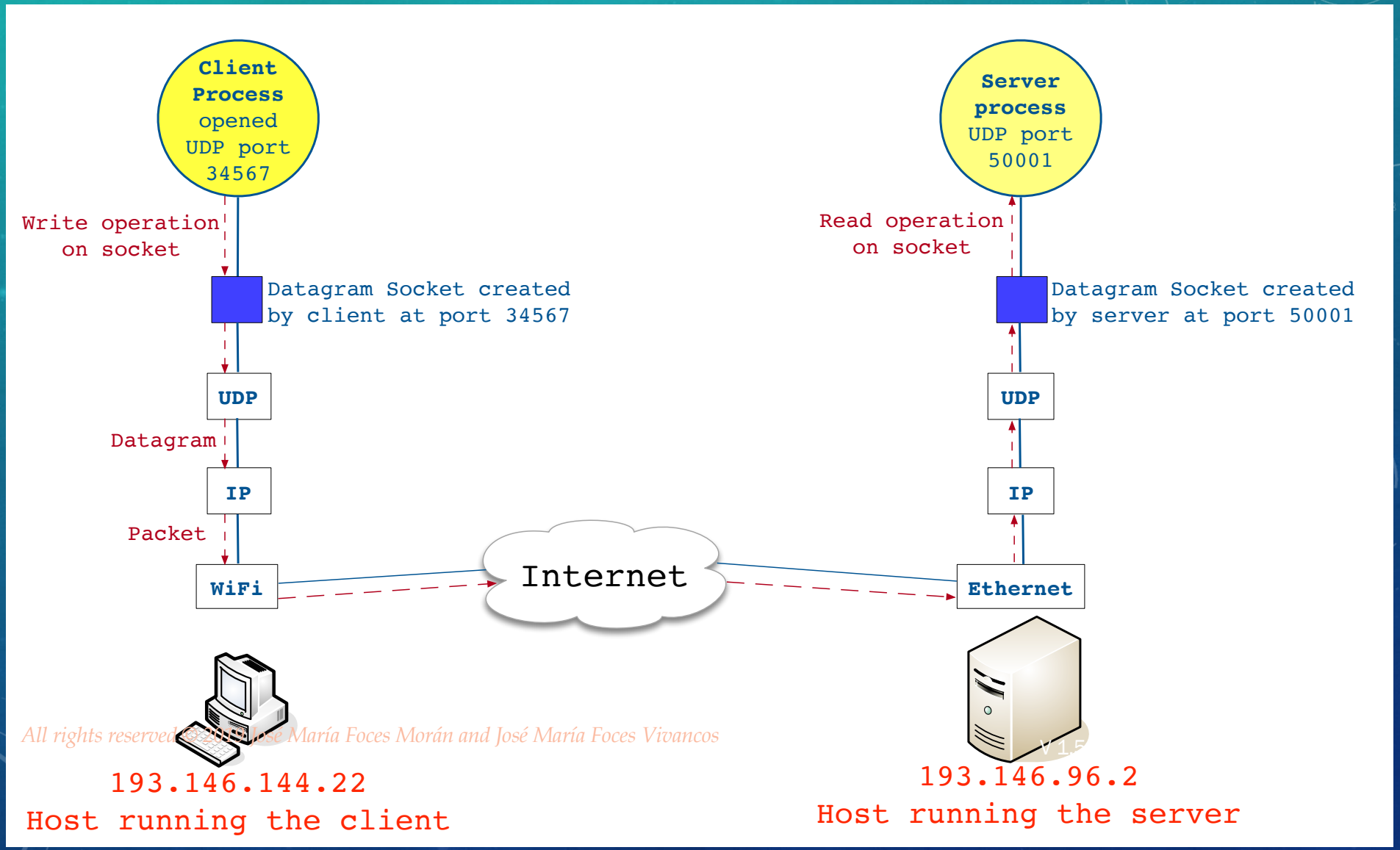
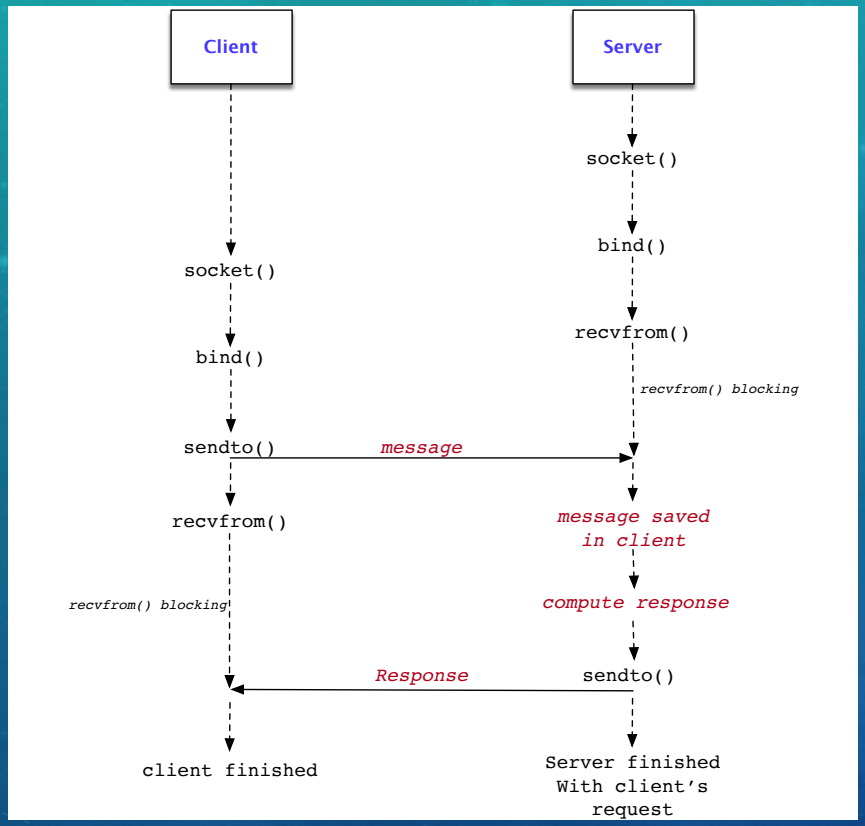
Sockets Interface and API

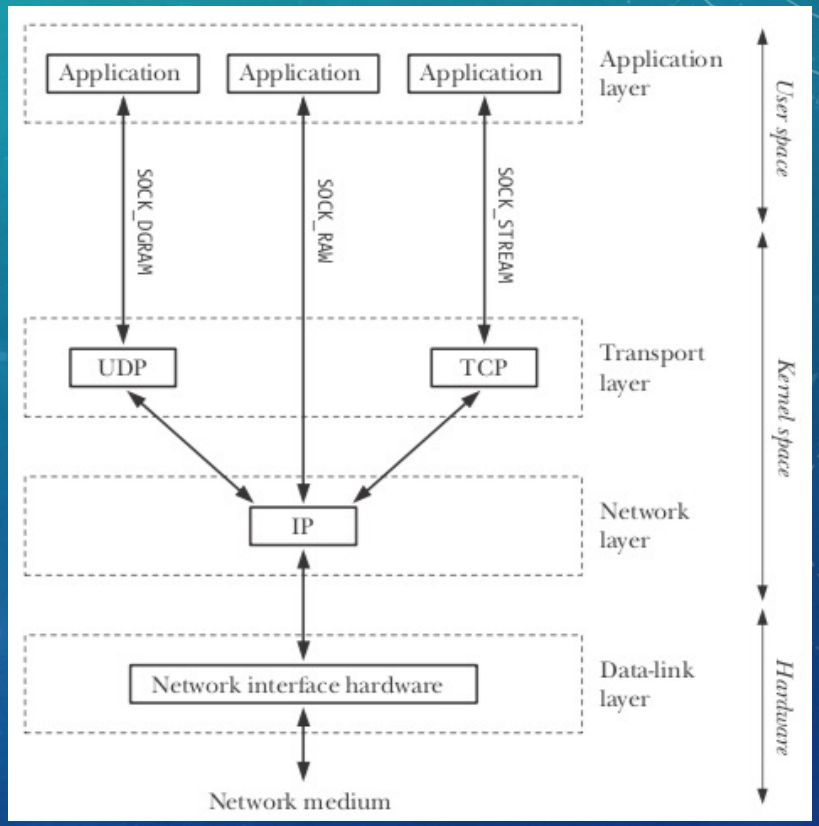
V 1.7 29/Sept/2023

Sockets: The service interface to Internet Architecture layers

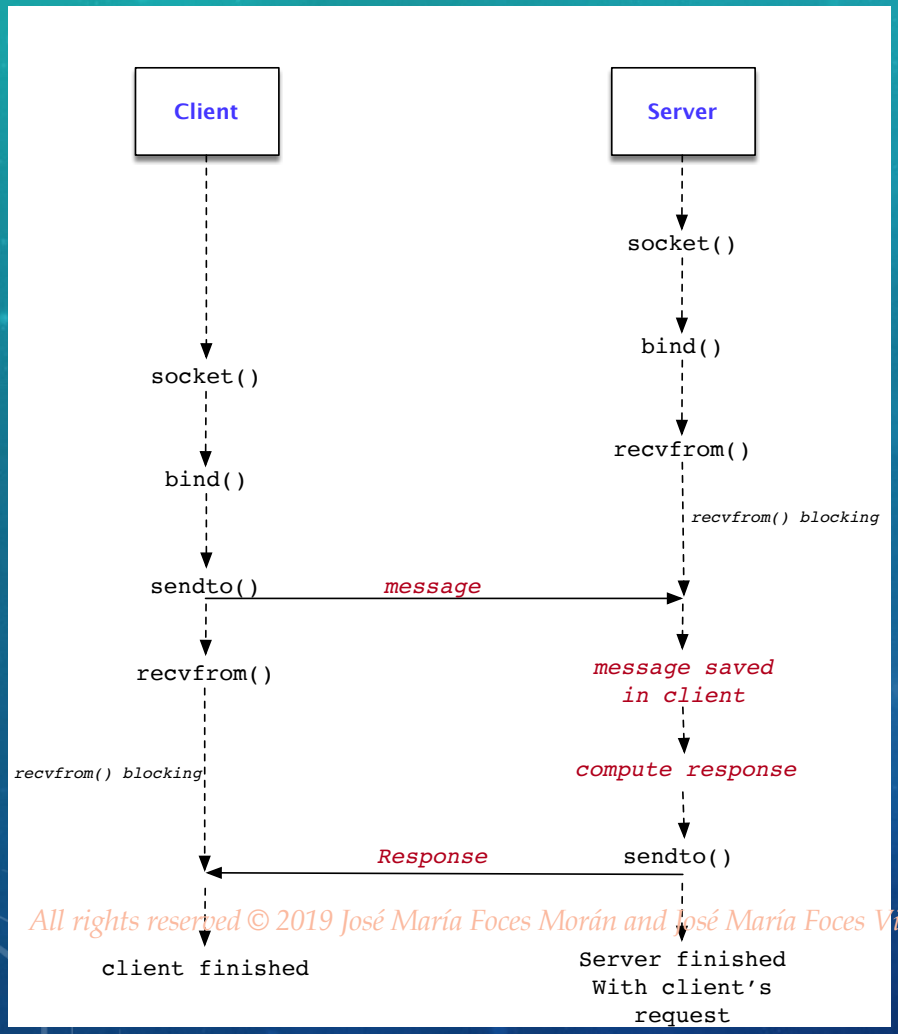


Sockets: The service interface to Internet Architecture layers



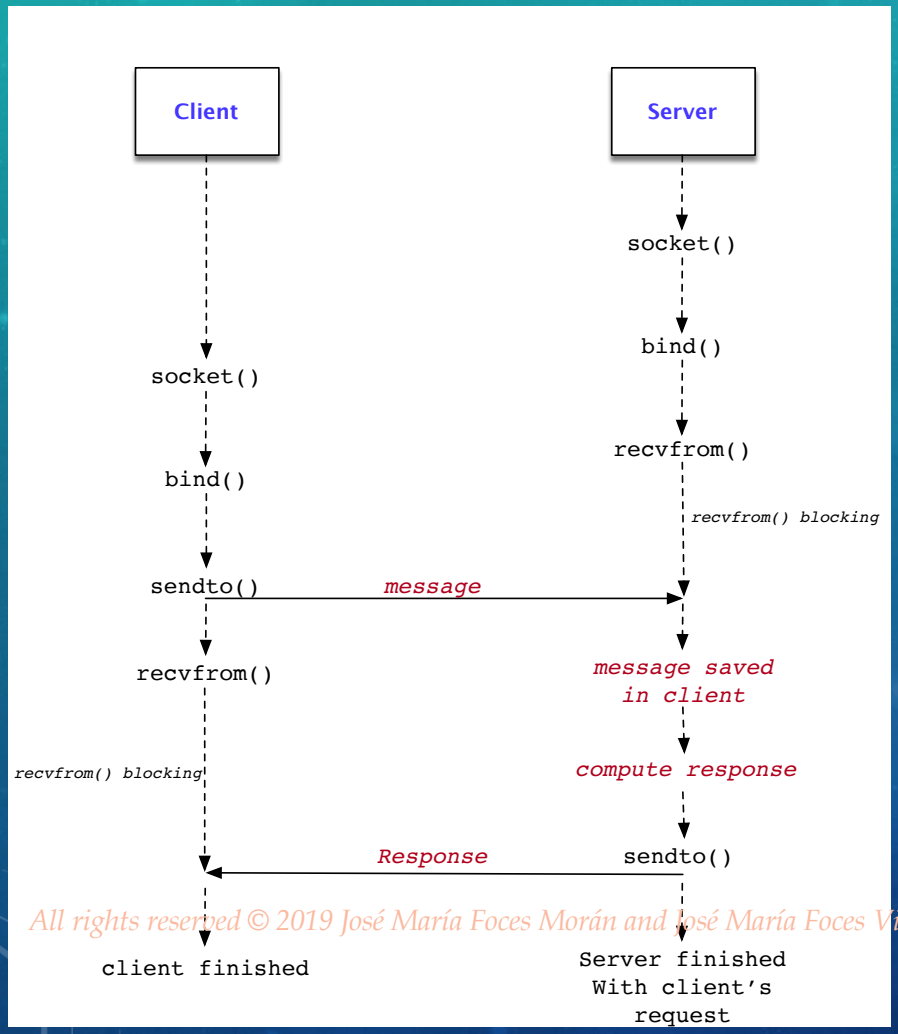


socket()




```
Create a Datagram socket (UDP):

#include <sys/socket.h>

fd = socket(domain, type, protocol);

      Domain: AF_INET; AF_INET6

      Type: SOCK_DGRAM, SOCK_STREAM

      Protocol: 0
```

bind()



Server

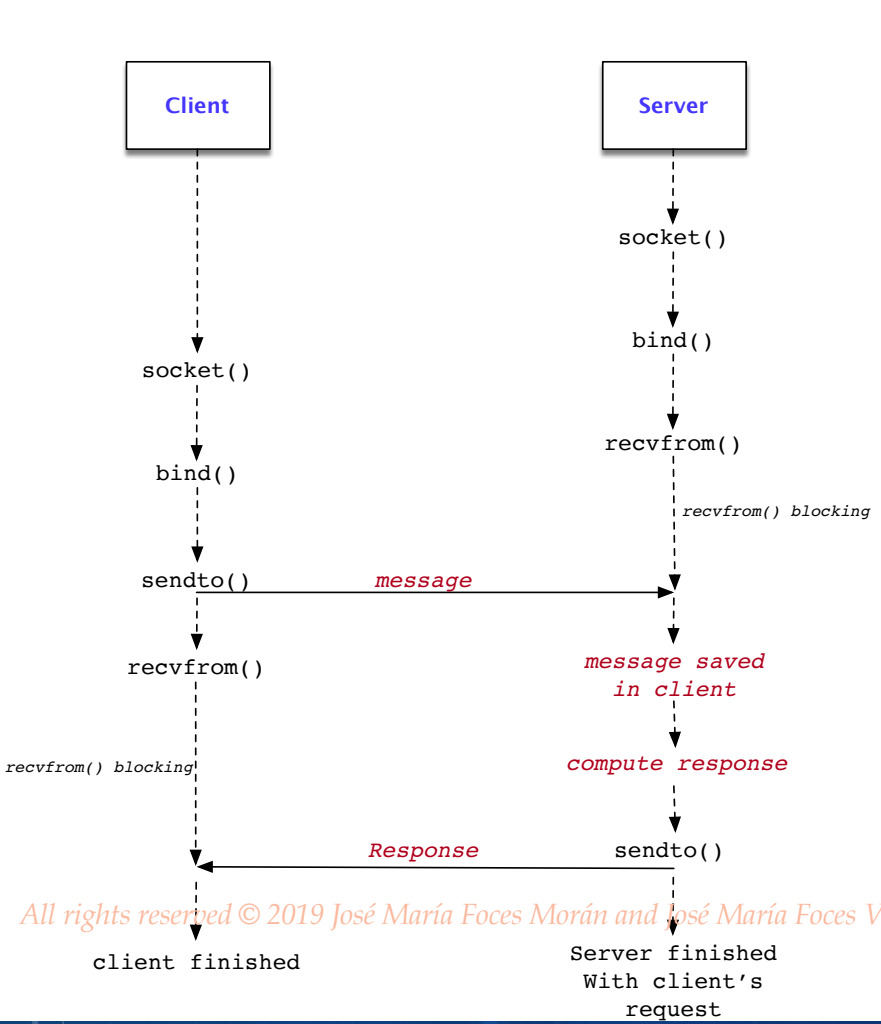
Bind the socket to a socket address:

· Server's IP address (193.146.96.2)

· Port number (50001)

```
#include <sys/socket.h>

int bind(int fd,
const struct sockaddr *addr,
socklen_t addrlen);
```

Inet Socket address, Inet address



Server

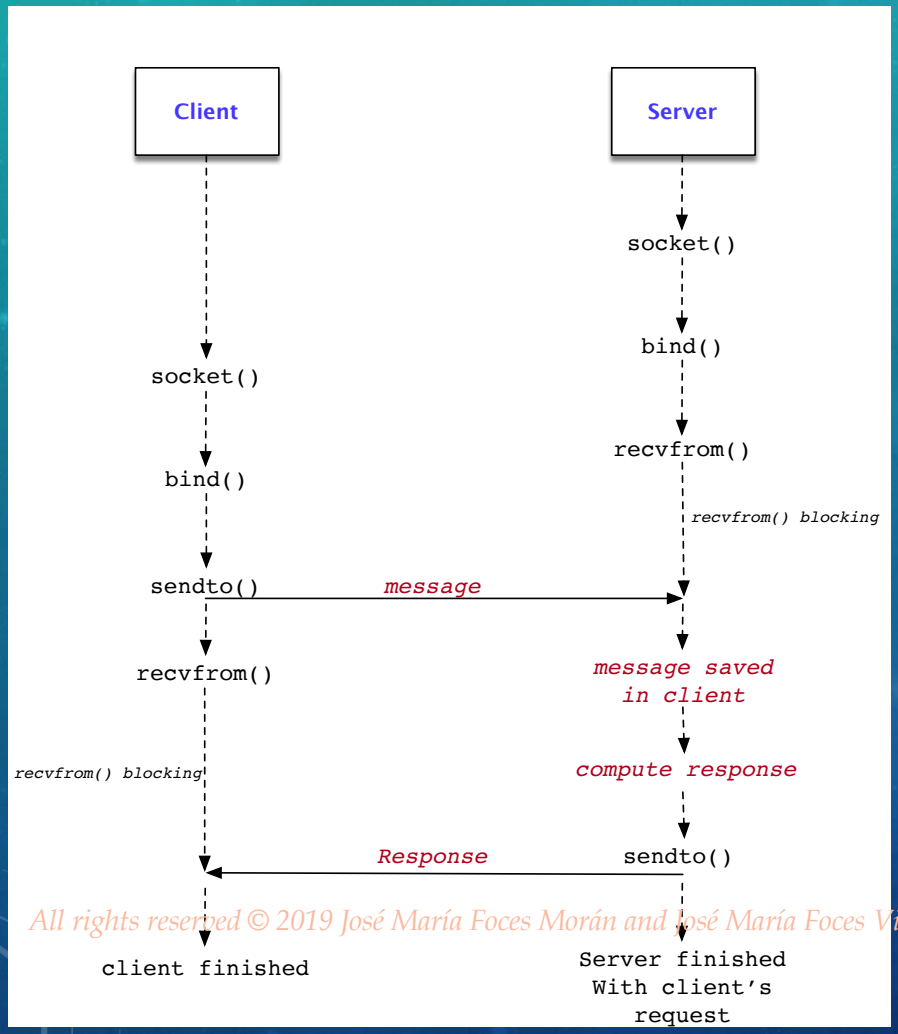
Bind the socket to a socket address:

· Server's IP address (193.146.96.2)

· Port number (50001)

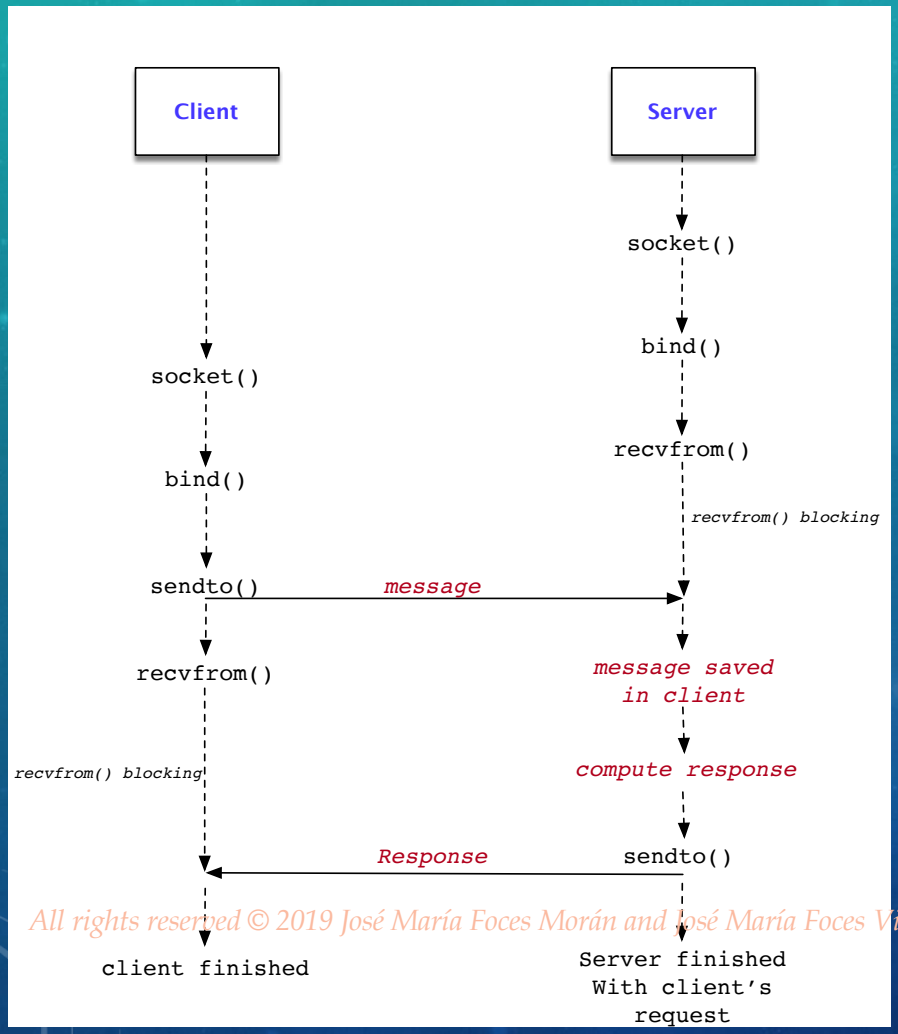
```
#include <netinet/in.h>
struct in_addr {
in_addr_t s_addr;
};
struct sockaddr_in {
sa_family_t sin_family;
in_port_t sin_port;
struct in_addr sin_addr;
unsigned char __pad[x];
};
```

Generic socket address **sockaddr** and Inet socketaddress **sockaddr_in**




```
-- A variety of socket-related calls
use generic socket addresses, wchi are
represented by struct sockaddr;


-- The specific socket address type
when the address family is AF_INET is
sockaddr_in

-- struct sockaddr and struct
sockaddr_in are not equal,
consequently, they cannot be assigned
to each other

-- Whenever we need to access a
sockaddr by using the fields in a
sockaddr_in, we can take a pointer to
sockaddr and have it cast into a
sockaddr_in:

struct sockaddr_in *p;

//f returns struct sockaddr

p = (struct sockaddr_in) f();
```

bind()



Server

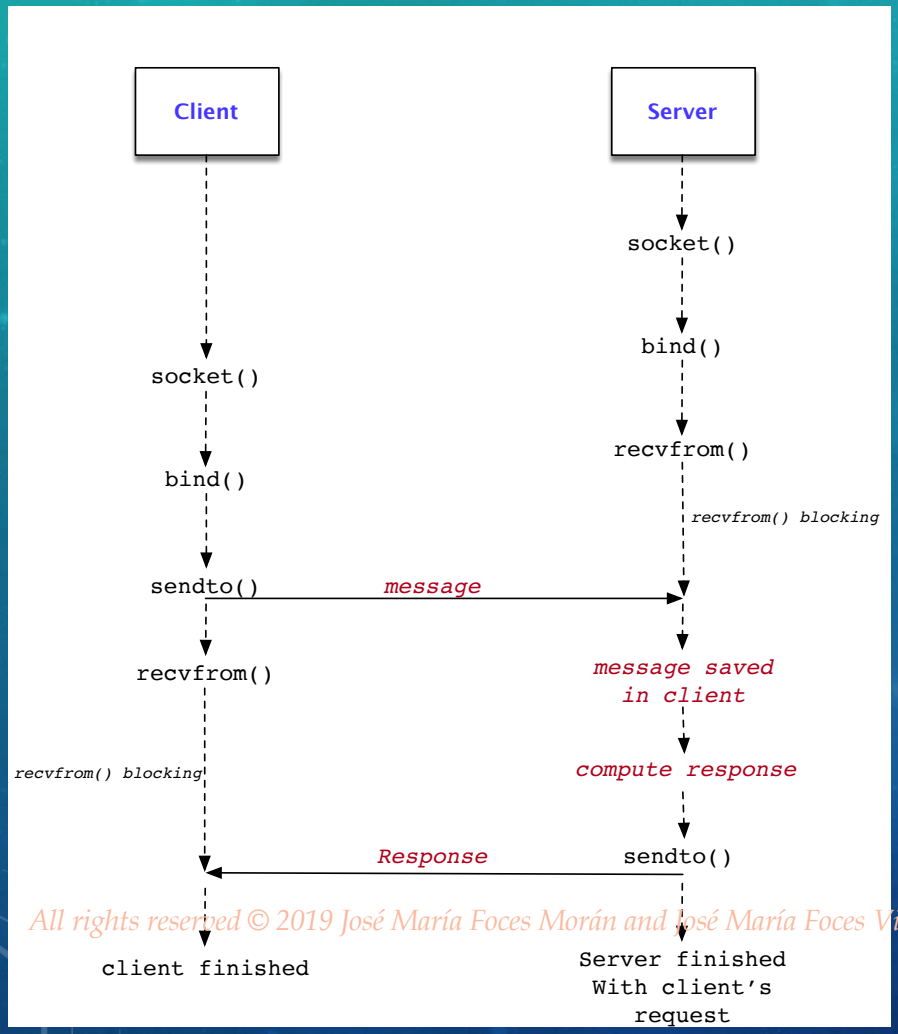
Bind the socket to a socket address:

· Server's IP address (193.146.96.2)

· Port number (50001)

```
#include <netinet/in.h>
struct sockaddr_in server;
server.sin_family = AF_INET;
server.sin_port = htons(50001);
server.sin_addr.s_addr = INADDR_ANY;
bind(
fd,
(struct sockaddr *) &server,
sizeof (server));
```

Socket address

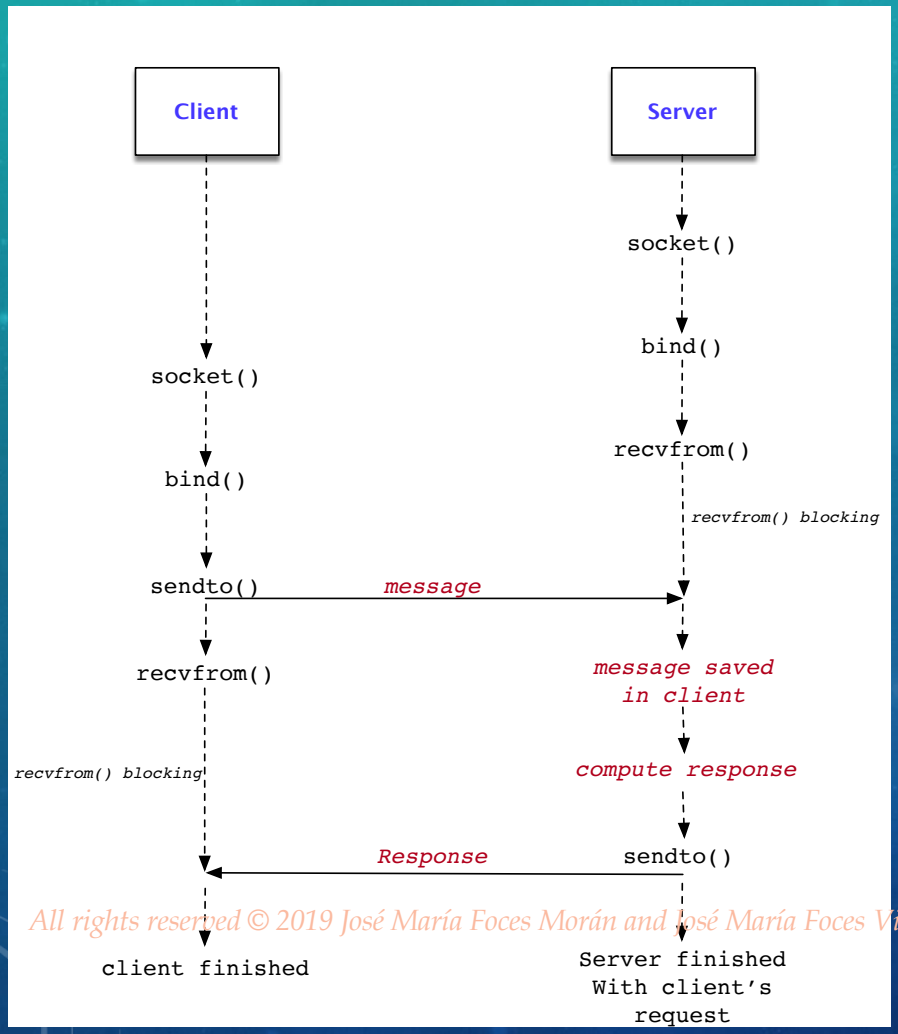


Client

Bind the socket any free local port and IP address:

```
-----------------------------------

#include <netinet/in.h>

struct sockaddr_in client;

    client.sin_family = AF_INET;

    client.sin_port = INADDR_ANY;

    client.sin_addr.s_addr = INADDR_ANY;
```

sendto()

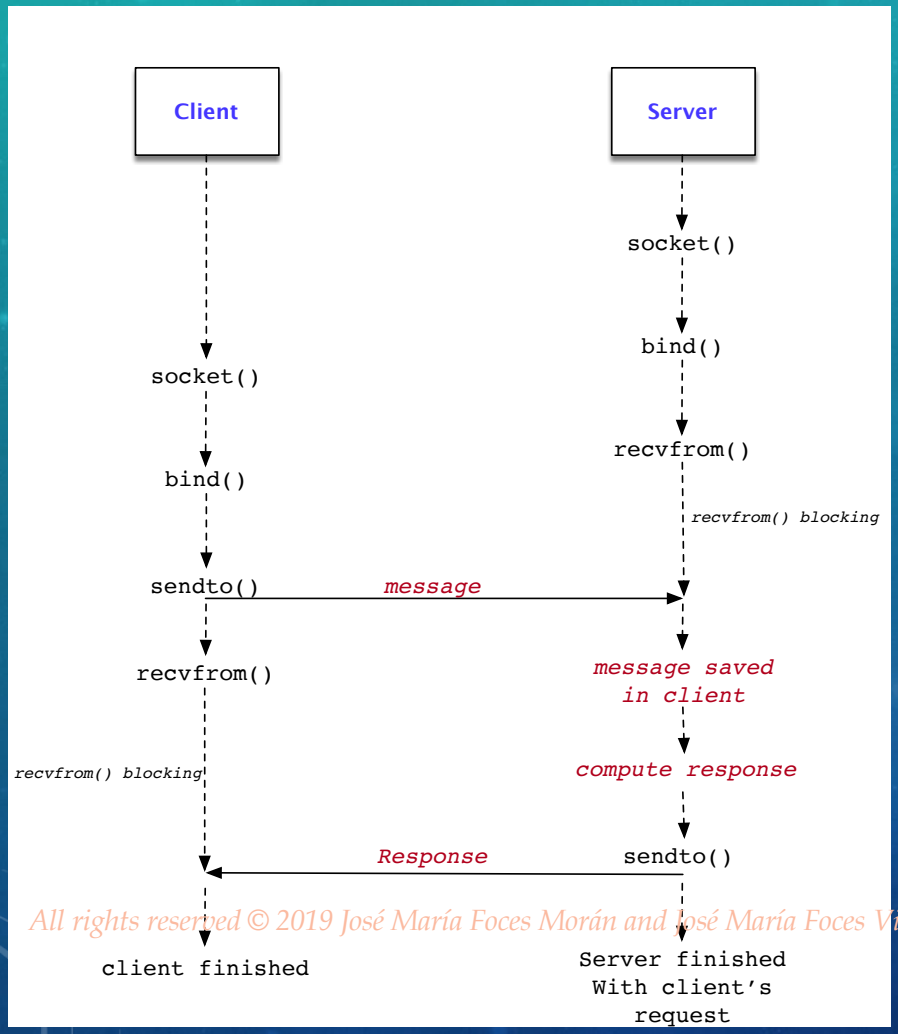


Client

Send message to server

```
struct sockaddr_in server;

server.sin_family = AF_INET;

server.sin_port = htons(50001);

server.sin_addr.s_addr =
inet_addr("193.146.96.2");


char *message = "Hello world! aeiou
abcdefeghijklmnopqrstuvxyz";

sendto(

fd,

message,

strlen(message),

0,

(struct sockaddr *) &server,

sizeof (server));
```

recvfrom()



Server

Receive message

```
struct sockaddr_in client;
unsigned int addr_length;

recvfrom(
fd,
message,
message_length,
0, //Flags
(struct sockaddr *) &client,
&addr_length
);
```

Server

```
/*
 * (C) José María Foces Morán
 * Computer Networks 2017
 * Basic, non-reentrant, UDP-based Echo Server
 *
 */

#include <stdio.h>
#include <stdlib.h>

#include <fcntl.h>
#include <memory.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>

#include <signal.h>
#include <errno.h>
#include <sys/time.h>

#define OURPORT 50001
#define REQLENGTH 127

char DATEREQUEST[REQLENGTH + 1];

int MAXITERATIONS = 5;
```

Server

```
32
33  int main(int argc, char** argv) {
34
35      /*
36       * Crear la dirección de la socket UDP
37       */
38      struct sockaddr_in server;
39      // Socket del tipo 'internet'
40      server.sin_family = AF_INET;
41      // Elegimos un puerto no reservado
42      server.sin_port = htons(OURPORT);
43      /* Esta socket (En este server) va a a escuchar en
44       * cualquiera de las IPs, reales o virtuales
45       */
46      server.sin_addr.s_addr = INADDR_ANY;
47
48      /*
49       * Se crea una socket internet, del tipo datagrama
50       */
51      int s;
52      s = socket(AF_INET, SOCK_DGRAM, 0); //Último arg, siempre 0
53
54
55      /*
56       * Unimos a la socket s la dirección que hemos creado en los
57       * párrafos anteriores:
58       */
59      bind(s, (struct sockaddr *) &server, sizeof (server));
60
61
62      /*
63       * Dirección del cliente que nos va a contactar
64       */
65      struct sockaddr_in client;
66      unsigned int addr_length;
67      //addr_length = sizeof (client);
68
69      printf("Servidor esperando a recibir una solicitud\n");
70      fflush(stdout);
71
72
```

Server

```
71
72
73      /*
74       * Usar la socket s para recibir un mensaje que se almacenará en DATEREQUEST y
75       * que no será de longitud efectiva superior a REQLENGTH (20 bytes),
76       * la dirección y el puerto del cliente quedarán registrados en client
77       */
78      addr_length = sizeof (client);
79      int i = 0;
80
81      while (i < MAXITERATIONS) {
82          int nbytes = recvfrom(s, DATEREQUEST, REQLENGTH, 0, (struct sockaddr *) &client, &addr_length);
83
84          printf("Solicitud recibida con un tamaño de %u bytes, procedentes de %s@ puerto %u:\n",
85                  nbytes, inet_ntoa(client.sin_addr), client.sin_port);
86
87          printf("'%s'\n", DATEREQUEST);
88
89          char *response = "Respuesta de prueba";
90
91          /*
92           * Usar la socket s para enviar un DATAGRAMA de respuesta al cliente desde
93           * el que hemos recibido la solicitud
94           */
95          nbytes = sendto(s, response, strlen(response) + 1, 0, (struct sockaddr *) &client, sizeof (client));
96
97          printf("Enviados %u bytes:\n\n", nbytes);
98
99          i++;
100     }
101 }
```

Client

```c
/*
 * (C) José María Foces Morán, Universidad de León
 *
 * Prácticas de Telecomunicaciones en la Industria 2017
 *
 * UDP Echo client: Sends a message 5 times and checks response from server
 *
 */

#include <stdio.h>
#include <stdlib.h>

#include <fcntl.h>
#include <memory.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>

#include <signal.h>
#include <errno.h>
#include <sys/time.h>
```

Client

```
27  int createDatagramSocket() {
28
29      /*
30       * Create a sockaddr_in struct that represents the full
31       * UDP socket addressing (IP and UDP port number)
32       */
33      struct sockaddr_in client;
34      client.sin_family = AF_INET;
35      client.sin_port = INADDR_ANY;
36      client.sin_addr.s_addr = INADDR_ANY;
37
38      /*
39       * Create a new UDP socket in the AF_INET domain and of
40       * type SOCK_DGRAM (UDP)
41       */
42      int s = socket(AF_INET, SOCK_DGRAM, 0); //Always 0
43
44      /*
45       * Bind the socket s to address client
46       */
47      bind(s, (struct sockaddr *) &client, sizeof (client));
48
49      return s;
50  }
```

Client

```
51
52  struct sockaddr_in createServerAddress() {
53
54      /*
55       * Create a sockaddr_in struct that represents the full
56       * UDP socket addressing for the server
57       */
58      struct sockaddr_in server;
59      server.sin_family = AF_INET;
60      server.sin_port = htons(50001);
61      //Type the server IP right below
62      server.sin_addr.s_addr = inet_addr("192.168.2.107");
63
64      return server;
65  }
```

Client

```c
void sendLoop(int s, struct sockaddr_in server) {

    int i = 0;

    while (i < 5) {

        printf("Send iteration %u\n", ++i);

        /*
         * Send mensaje through socket s to UDP server socket
         * whose address is server
         *
         * flags is 0
         */
        char *message = "Hello world! aeiou abcdefeghijklmnopqrstuvxyz";

        int nbytes = sendto(s, message, strlen(message), 0, (struct sockaddr *) &server, sizeof (server));

        printf("%u actually sent\n", nbytes);
        printf("%s\n", message);

        fflush(stdout);

        char response[1025];

        struct sockaddr_in addr;
        unsigned int addr_length;

        addr_length = sizeof (addr);
        nbytes = recvfrom(s, response, 1024, 0, (struct sockaddr *) &addr, &addr_length);

        printf("%u bytes received:\n", nbytes);
        response[nbytes] = '\0';
        printf("%s\n", response);
        sleep(3);
    }

}
```

Client

```
106 int main(int argc, char** argv) {
107
108
109     int s = createDatagramSocket();
110
111     struct sockaddr_in server = createServerAddress();
112
113     sendLoop(s, server);
114
115     printf("Client exiting\n");
116
117 }
```