

Sockets in C

Gabriel de Biasi

biasi@dcc.ufmg.br

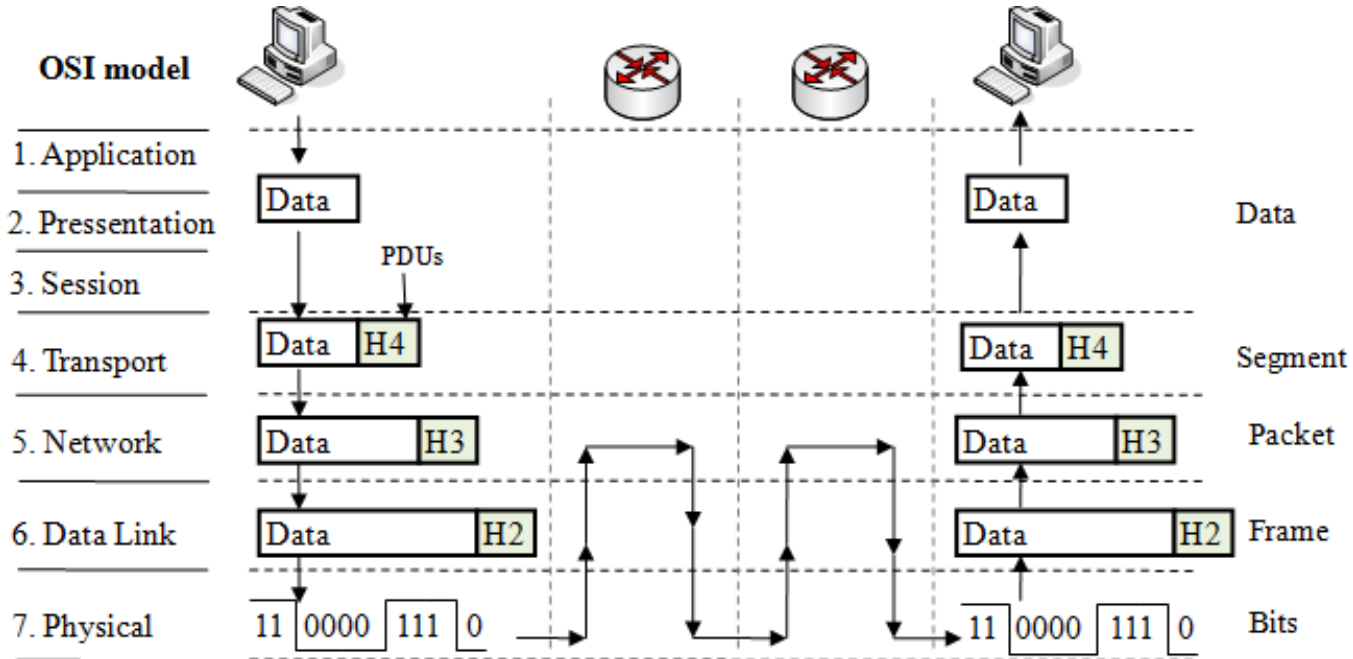
Department of Computer Science

Federal University of Minas Gerais, Brazil



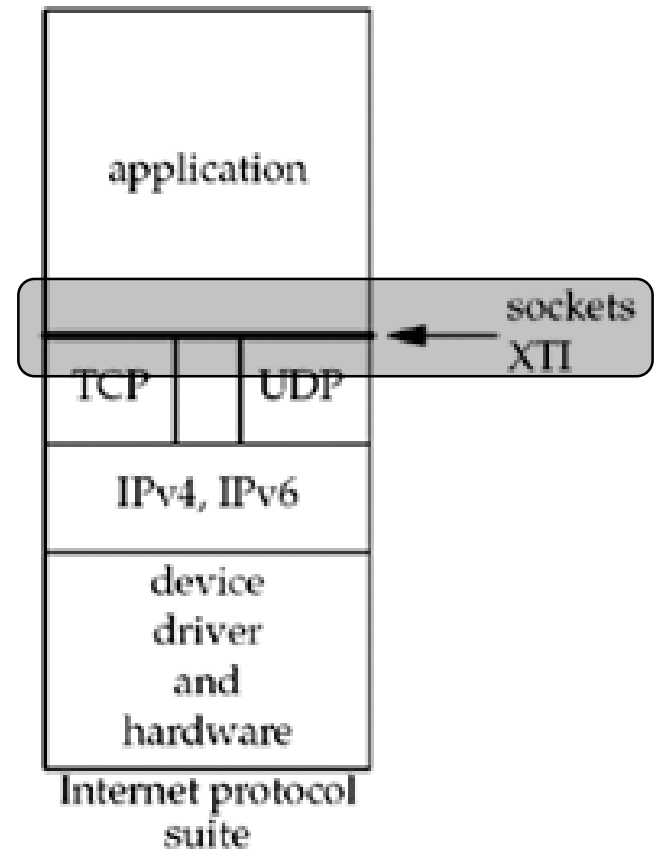
OSI Model

A conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system.





Sockets

Sockets are “bridges” between the application layer and the transport layer.



Types of sockets in C

- Stream Sockets (TCP):
 - Reliable;
 - Sequentially;
 - Connection-oriented; 
- Datagram Sockets (UDP):
 - Packets might be received out of order;
 - Packets might be lost;
 - Not connection-oriented; 

Behavior of sockets in C

- Servers:
 - Wait for new connections;
 - Provide services for clients;
- Clients:
 - Request a connection to server;
 - Request services provided by the server;

Functions

Function	Description
socket()	Creates a new socket;
bind()	Attach a IP address and a port to a socket;
listen()	Put the socket on passive mode, waiting for connections;
connect()	Starts a connection with the server;
accept()	Blocks the server until a new connection arrives;
send()	Send a message through the socket;
recv()	Receive a message from the socket;
sendto()	Send a message specifying the address;
recvfrom()	Receive a message and store the sender data;
close()	Close the connection.

Creating sockets in C

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

Family	Types	Protocol
AF_INET (IPv4)	SOCK_STREAM (TCP)	0 (default)
AF_INET6 (IPv6)	SOCK_DGRAM (UDP)	

The struct “sockaddr_in”

Struct used for both server and client, in order to set the IP address and port configurations.

```
struct sockaddr_in {  
    short          sin_family;           AF_INET  
    struct in_addr sin_addr;            IP Address  
    u_short        sin_port;            Port  
    char           sin_zero[8];         *Ignore*  
};
```


Server Functions in C

Attaching IP Address and Port to the socket:

```
bind(int socket_id, struct sockaddr* server_addr, int addr_len);
```

Putting in passive mode:

```
listen(int socket_id, int backlog);
```

Blocks the process until a new connection arrives:

```
accept(int socket_id, struct sockaddr* client_addr, int* addr_len);
```

Client Functions in C

Client starting a connection:

- `connect(int socket_id, struct sockaddr* server_addr, int addr_len);`

Closing the connection:

- `close(int socket_id);`

Exchanging Messages in C

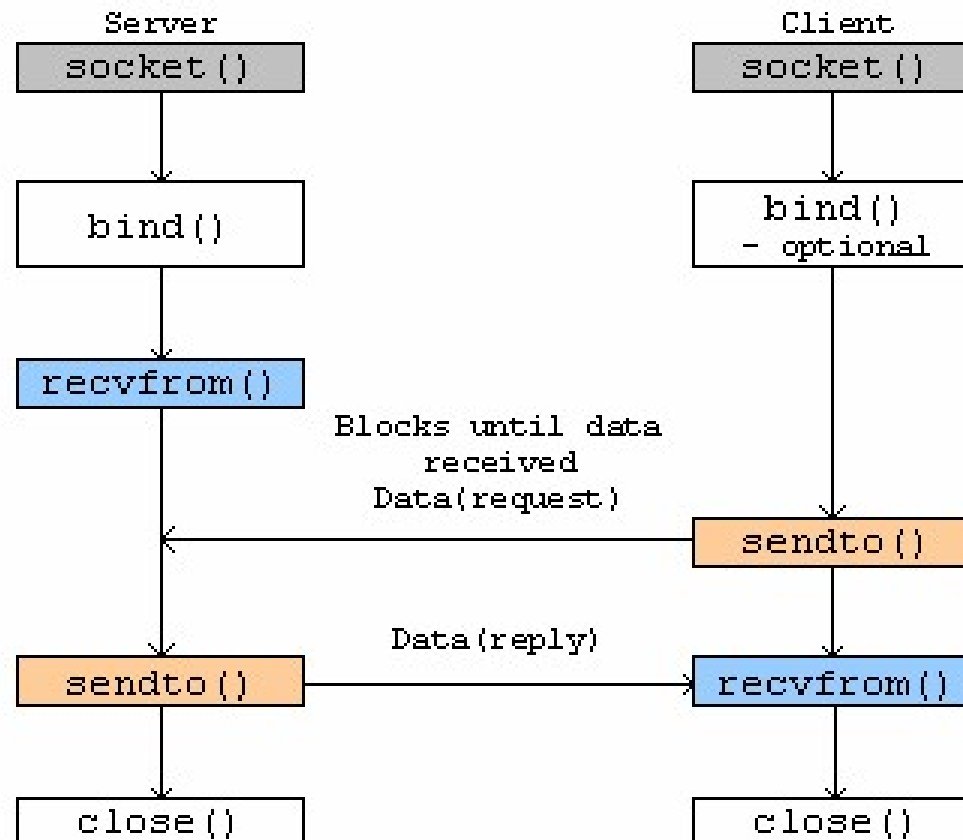
Sending message:

- `send(int socket_id, char* message, int message_len, int flags);`

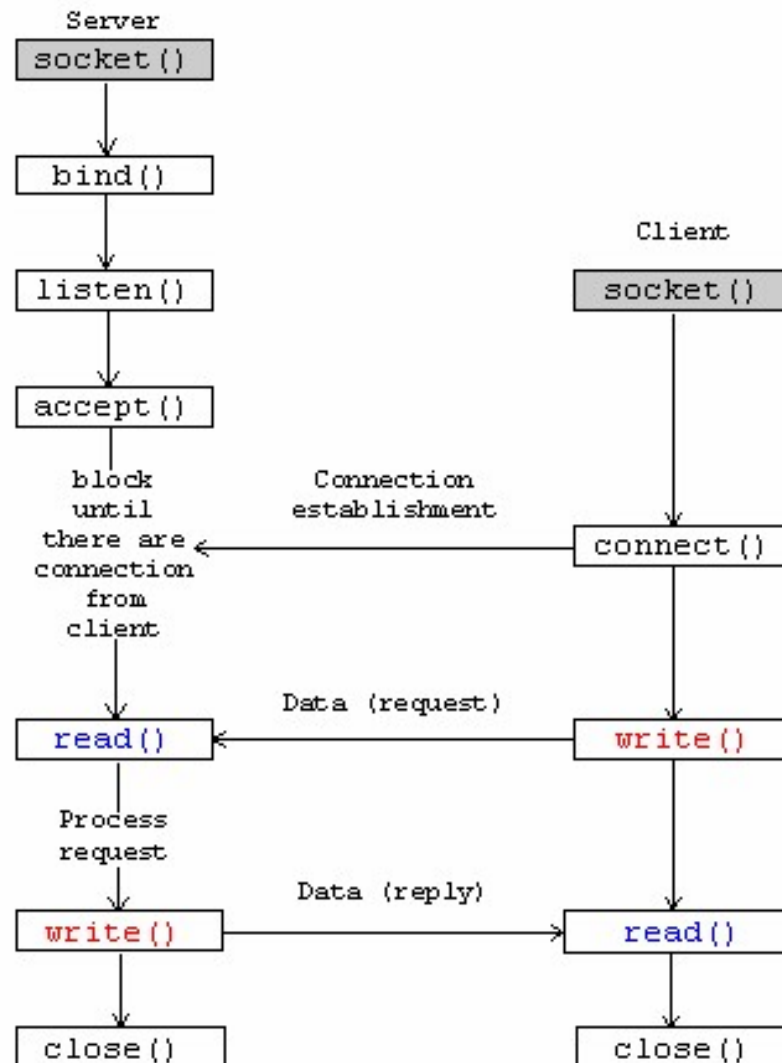
Receive message:

- `recv(int socket_id, char* message, int message_len, int flags);`

Sockets for UDP Protocol using C



Sockets for TCP Protocol using C



TCP and UDP Examples

Available: <http://dcc.ufmg.br/~biasi>

Teaching -> Computer Networks -> Examples

UDP Example: Simple Client-Server

A simple UDP client-server application that whenever the client sends a text to the server, it's return the same text with caps letters.

Piece of a cake!

TCP Example: Another simple Client-Server

A simple TCP client-server application that whenever the client connects to the server, it's able to send two integer numbers and the server returns the sum of them until the client send zero, that means to end the program.

Note: In this case, the server will remain blocked in to a single client until the connection is closed. There are methods for avoid this using parallel programming, but that is not the aim for this lecture.

Sockets IPv6

Creating a socket to use IPv6 communication:

- `int socket_id = socket(AF_INET6, int type, int protocol);`

The other functions doesn't change the syntax!

Sockets IPv6

Creating a socket to use IPv6 communication:

- `int socket_id = socket(AF_INET6, int type, int protocol);`

The other functions doesn't change the syntax!

The struct “sockaddr_in6”

Struct used for both server and client, in order to set the IP address and port configurations for IPv6 communications.

```
struct sockaddr_in6 {  
    sa_family_t    sin6_family;    AF_INET6  
    in_port_t      sin6_port;      Port  
    uint32_t       sin6_flowinfo;  Flow Info  
    struct in6_addr sin6_addr;      IP Address  
    uint32_t       sin6_scope_id;   Scope ID  
};
```

Discussion

Thank you!

Gabriel de Biasi

biasi@dcc.ufmg.br

Department of Computer Science

Federal University of Minas Gerais, Brazil

