Introduction

There are two primary transport layer protocols for host-to-host communication: TCP and UDP. Unlike TCP, where the client establishes a connection with the server, UDP operates without forming a connection. Instead, the client sends a datagram directly. The server, in turn, does not need to accept a connection but simply waits to receive datagrams. Each incoming datagram includes the sender's address, which the server uses to send data back to the appropriate client.

Socket

A socket is a combination of IP address and port on one system. On each system a socket exists for a process interacting with the socket on other system over the network.

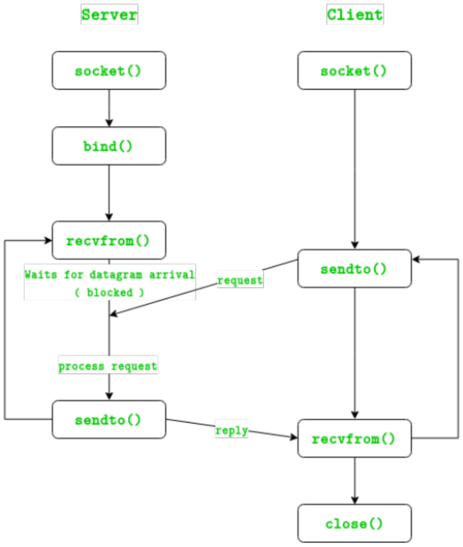
Function Descriptions

socket()
Creates an UN-named socket inside the kernel and returns an integer known as socket descriptor.
bind()
Assigns the details specified in the structure ‘serv_addr’ to the socket created in the step above.
listen()
With second argument as ’10’ specifies maximum number of client connections that server will queue for this listening socket.
accept()
The server is put to sleep and when for an incoming client request, the three-way TCP handshake is complete, the function accept () wakes up and returns the socket descriptor representing the client socket.

Algorithm :

```
server()
create UDP socket
Bind socket to address
wait for datagram from client
process and reply to client request
repeat while server is active
```

```
client()
create UDP socket
send request to server
wait for datagram from server
process and reply from server
close socket and exit
```


Server
Client
socket()
socket()
bind()
recvfrom()
sendto()
Waits for datagram arrival
(blocked)
request
process request
sendto()
reply
recvfrom()
close()