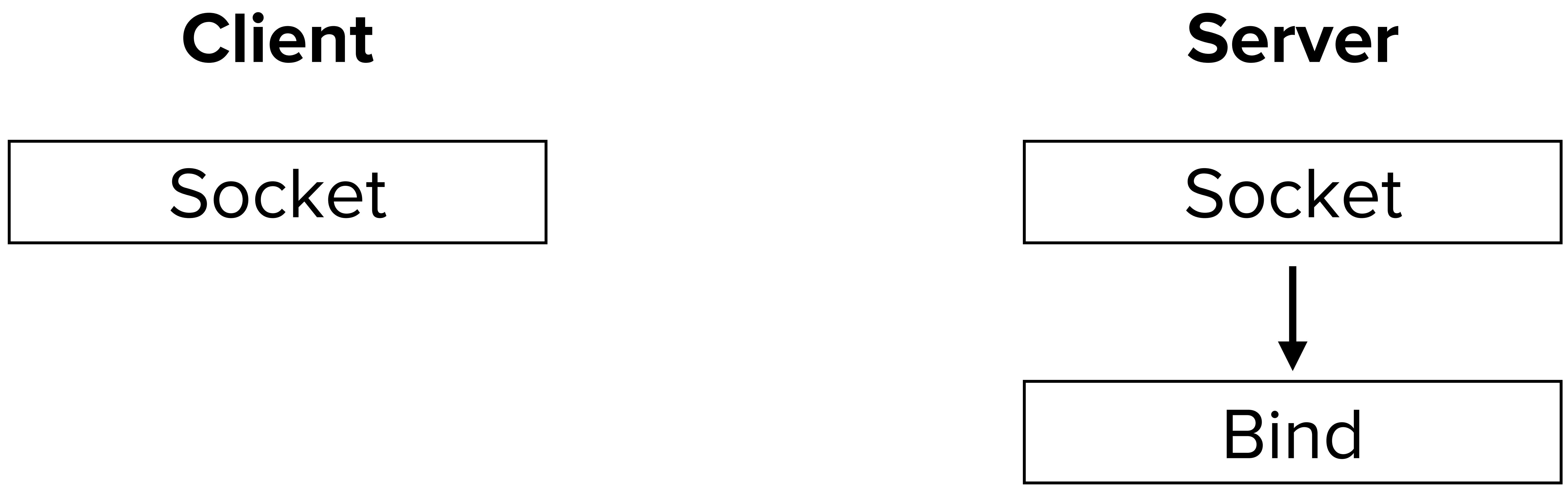
Computer Networks

week 6. Sockets. TCP

Client

Server

Client	Server
Socket	Socket



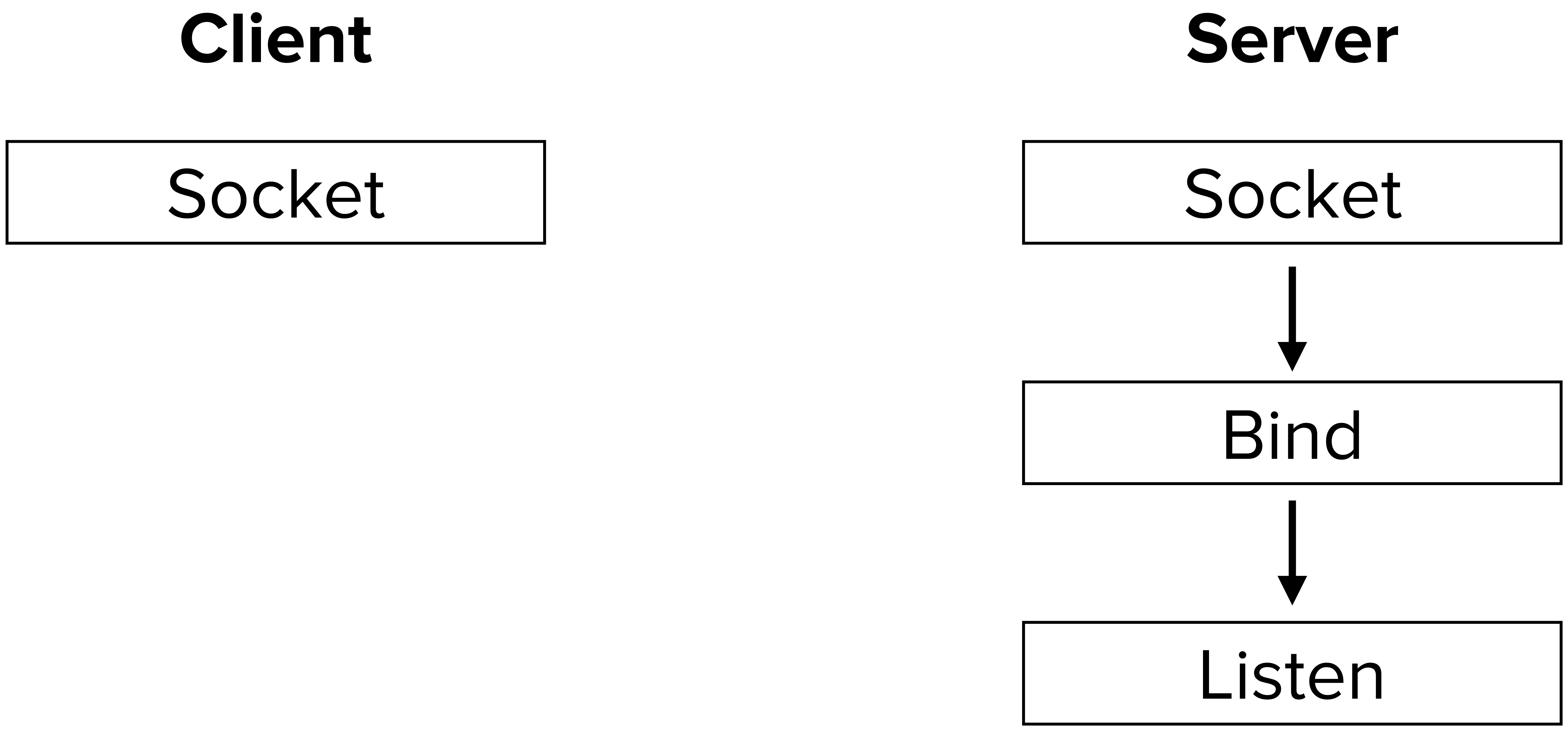
Client

Socket

Server

Socket

Bind



Client

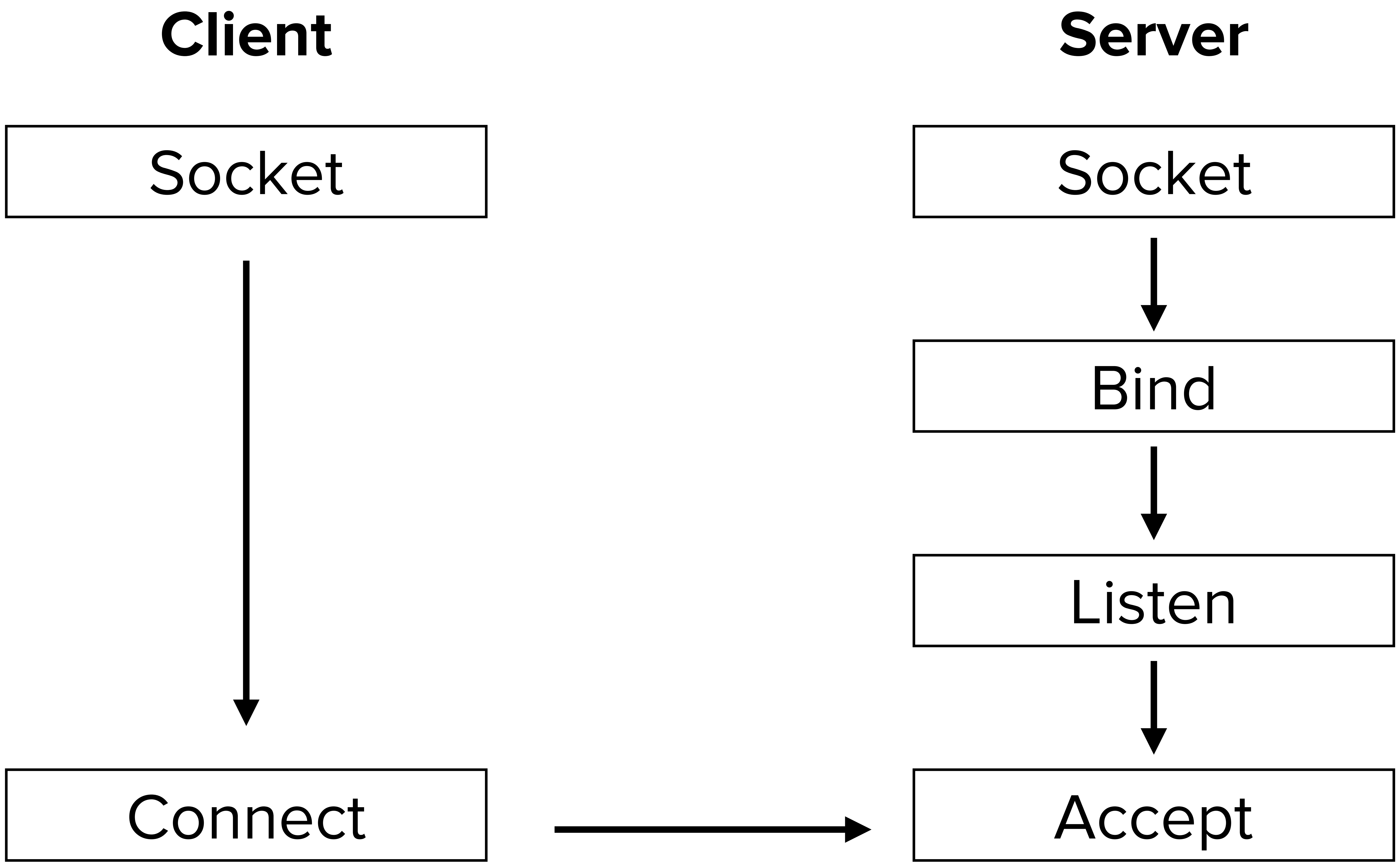
Socket

Server

Socket

Bind

Listen



Client

Socket

Connect

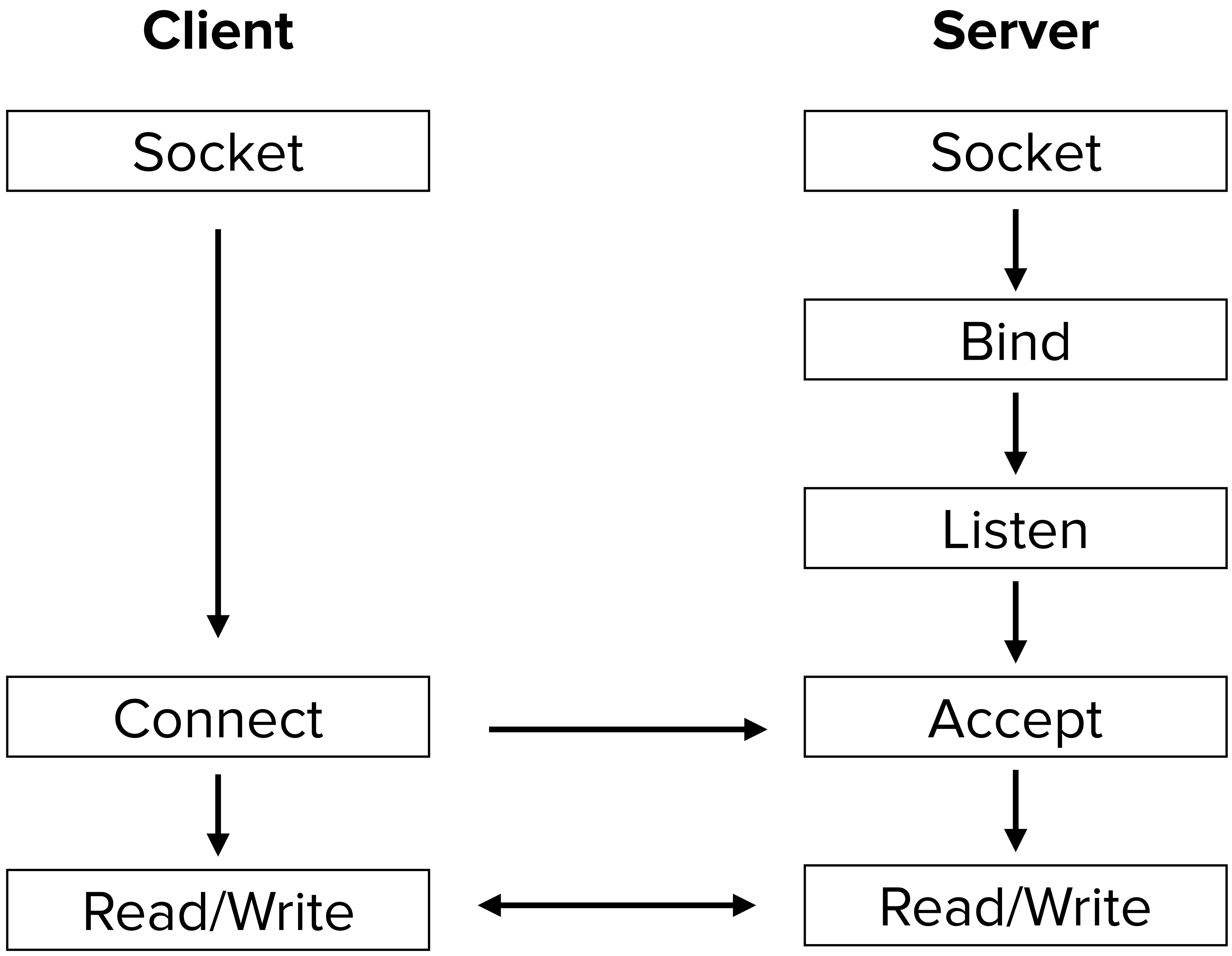
Server

Socket

Bind

Listen

Accept



Client

Socket

Connect

Read/Write

Server

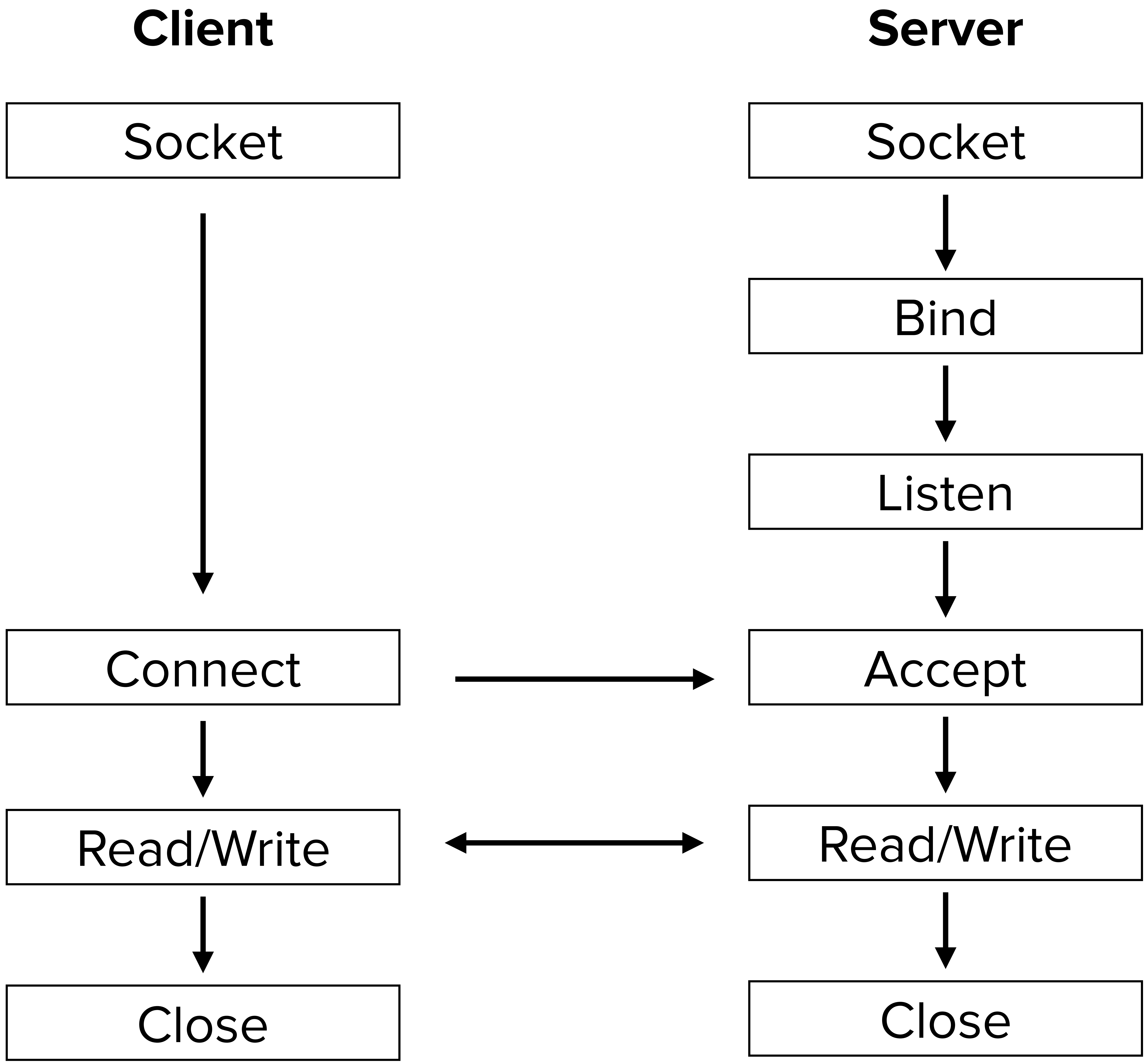
Socket

Bind

Listen

Accept

Read/Write



Client

Socket

Connect

Read/Write

Close

Server

Socket

Bind

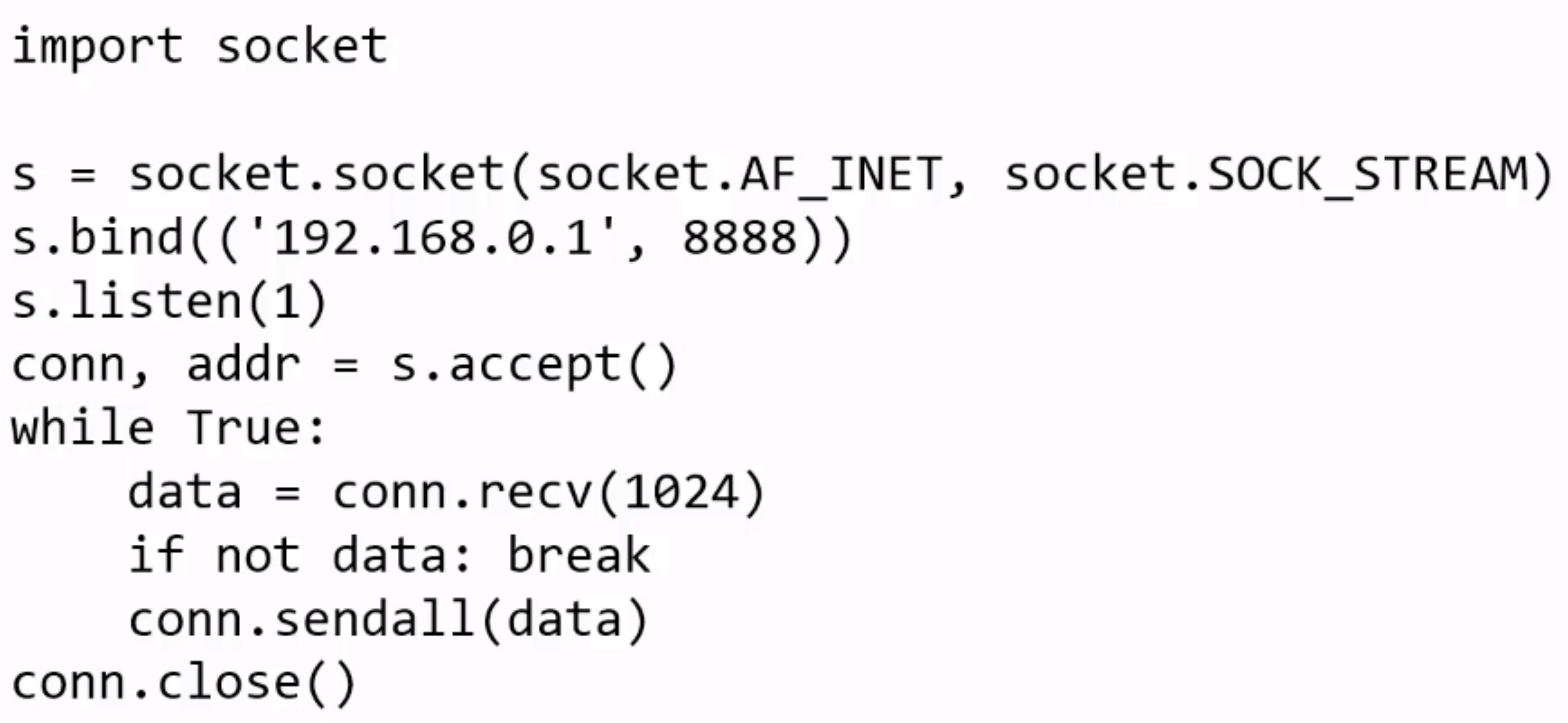
Listen

Accept

Read/Write

Close

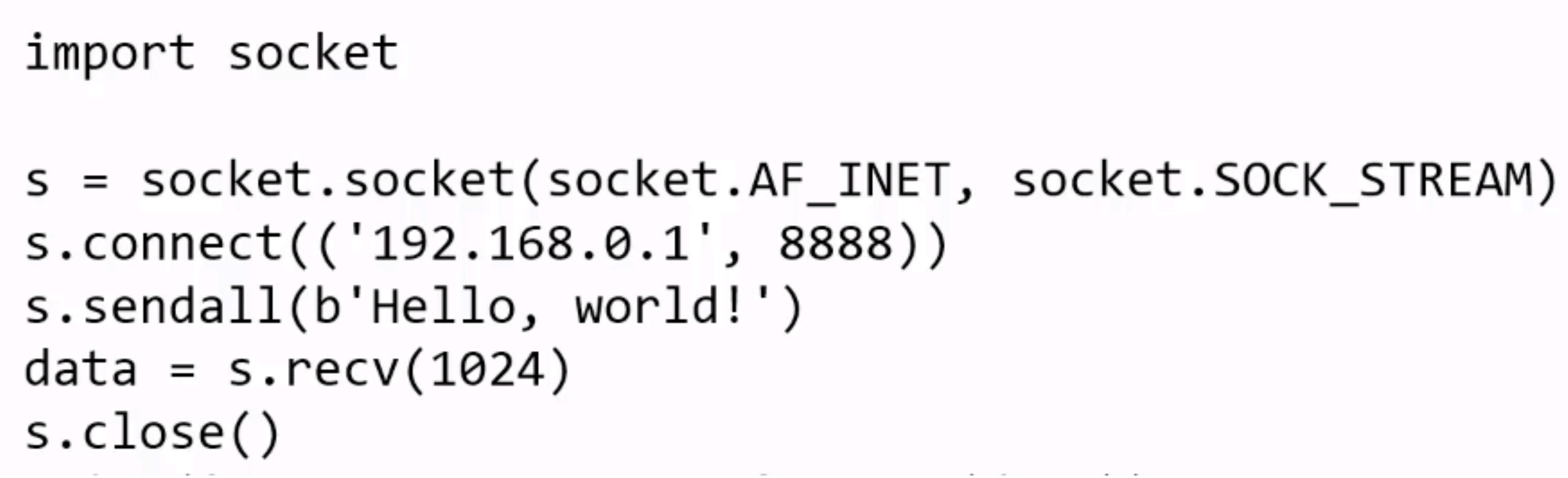
Server side code example (python)



```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('192.168.0.1', 8888))
s.listen(1)
conn, addr = s.accept()
while True:
    data = conn.recv(1024)
    if not data: break
    conn.sendall(data)
conn.close()
```

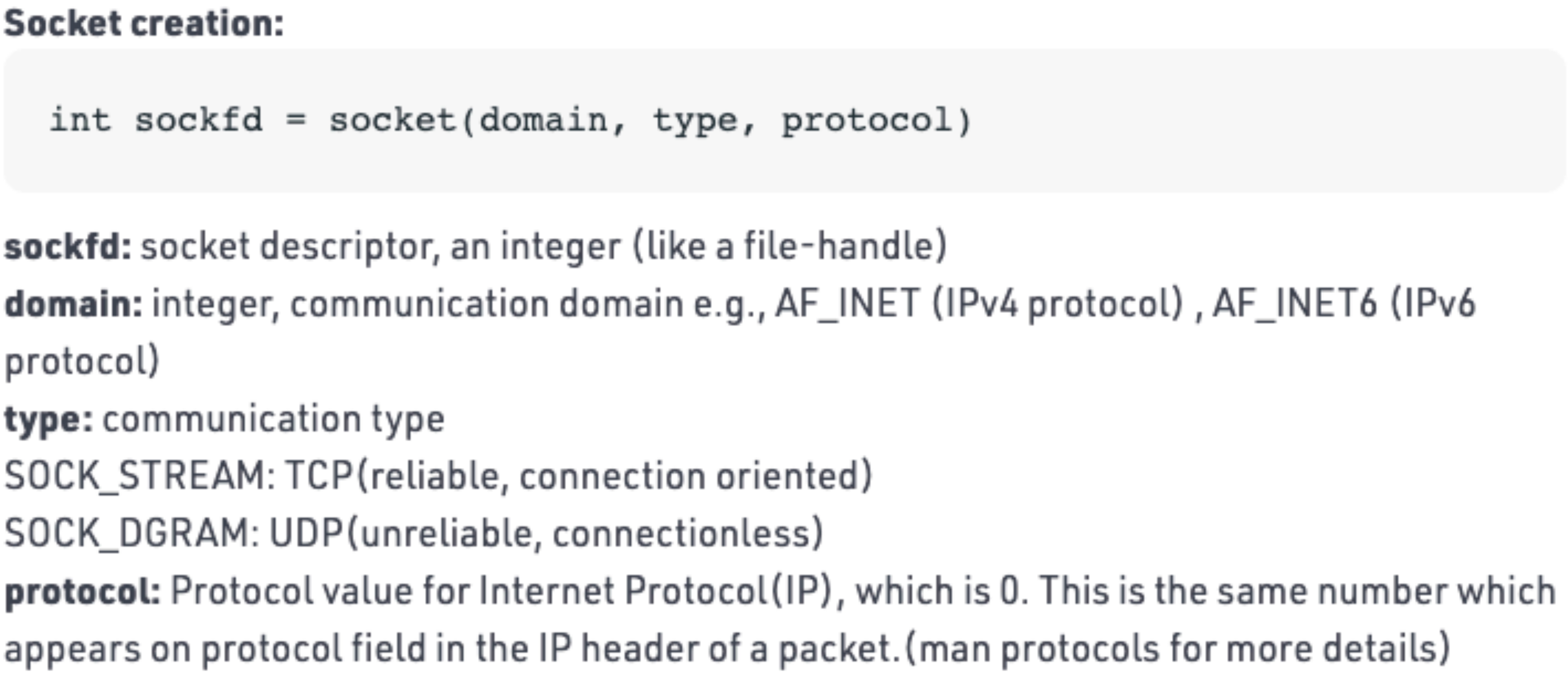
Client side code example (python)



```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.0.1', 8888))
s.sendall(b'Hello, world!')
data = s.recv(1024)
s.close()
```

Stages for server (C language)

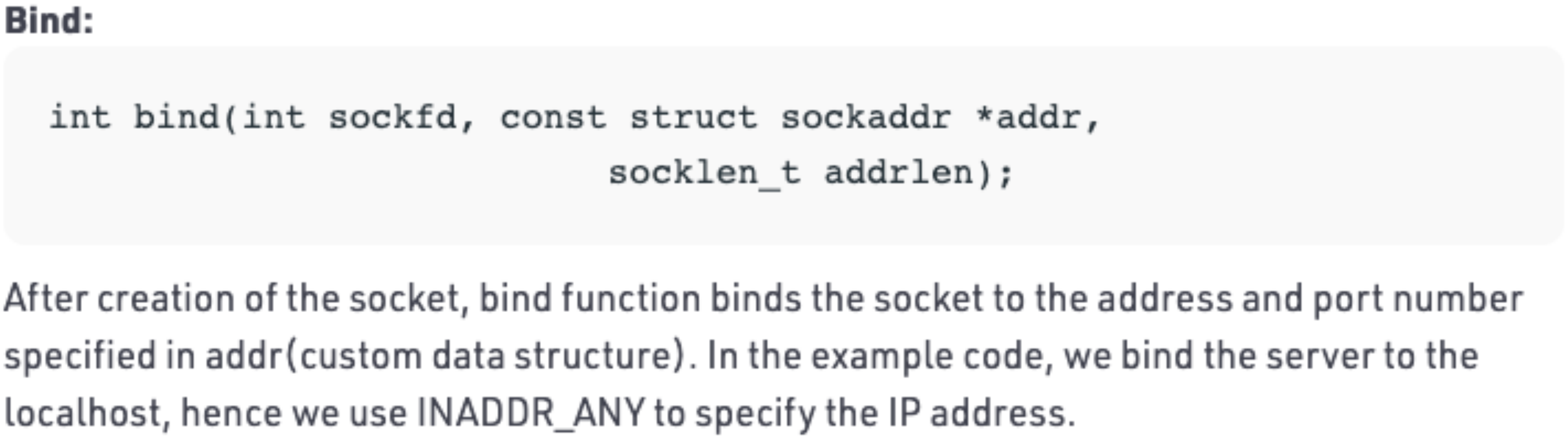


Socket creation:

```
int sockfd = socket(domain, type, protocol)
```

sockfd: socket descriptor, an integer (like a file-handle)
domain: integer, communication domain e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol)
type: communication type
SOCK_STREAM: TCP(reliable, connection oriented)
SOCK_DGRAM: UDP(unreliable, connectionless)
protocol: Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

Stages for server (C language)

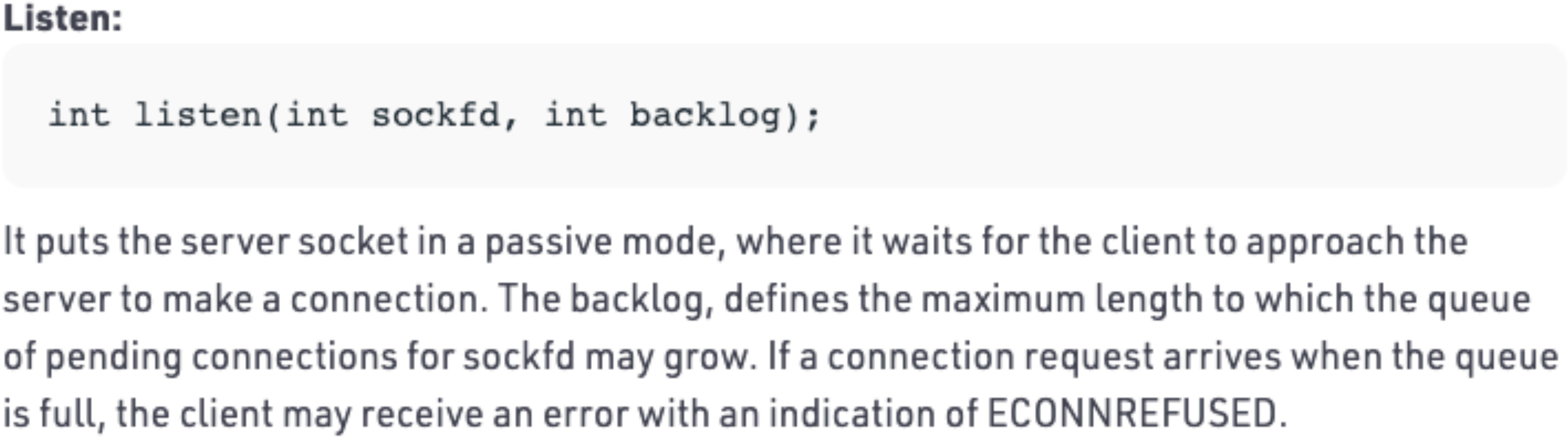


Bind:

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

After creation of the socket, bind function binds the socket to the address and port number specified in addr(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address.

Stages for server (C language)

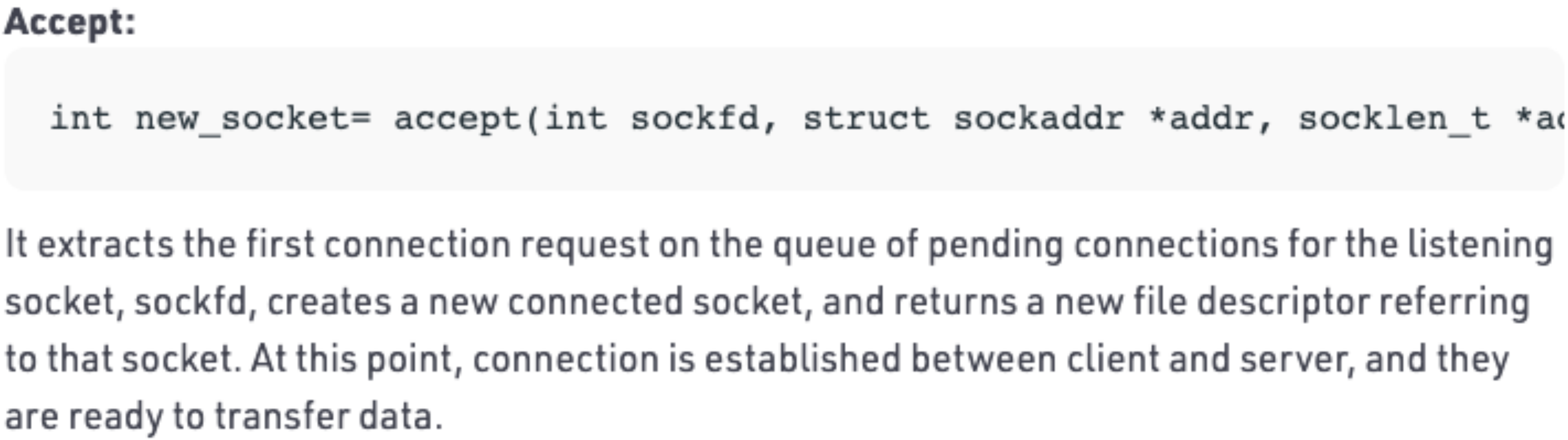


Listen:

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

It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

Stages for server (C language)

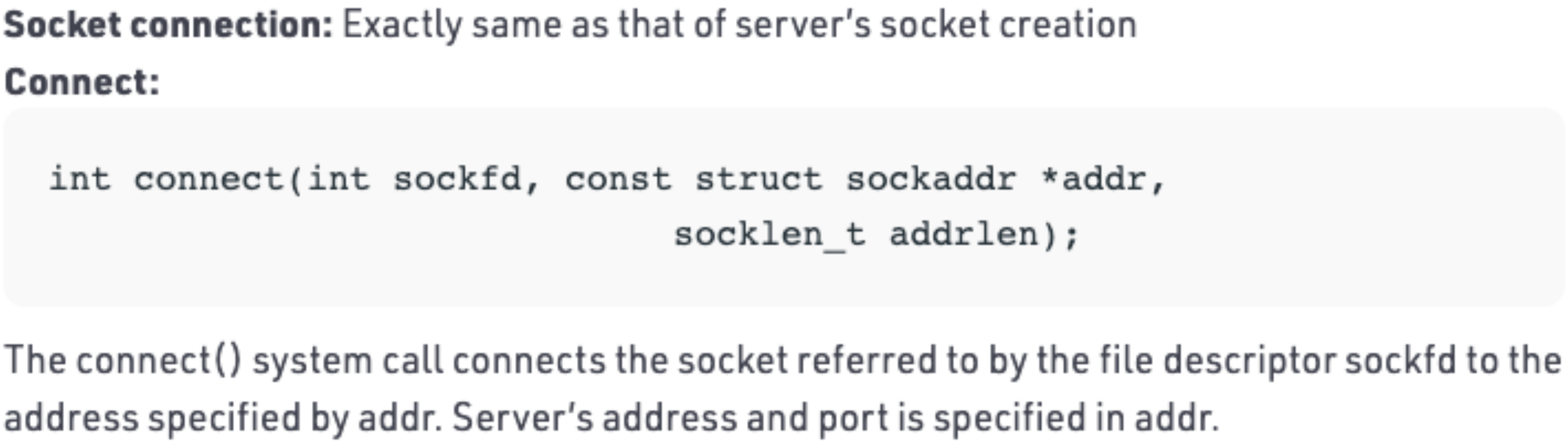


Accept:

```
int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *ad
```

It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.

Stages for client (C language)



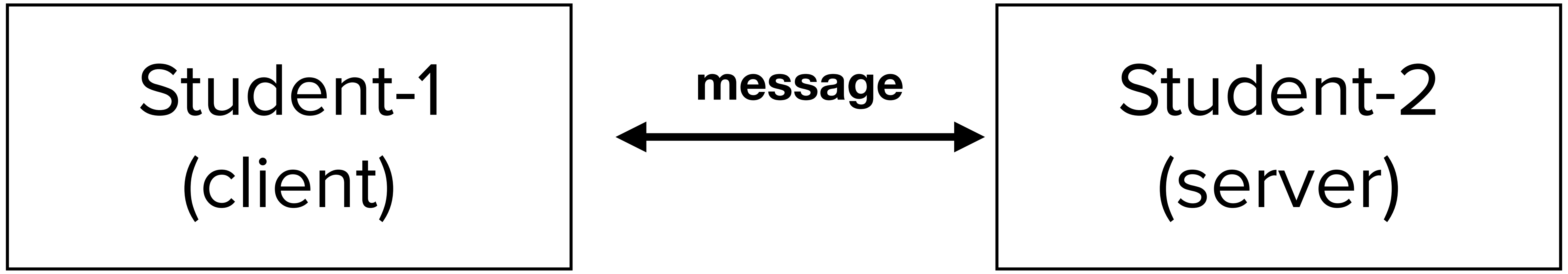
Socket connection: Exactly same as that of server's socket creation

Connect:

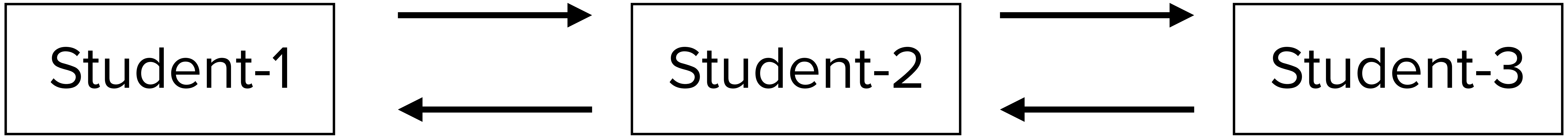
```
int connect(int sockfd, const struct sockaddr *addr,
                              socklen_t addrlen);
```

The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server's address and port is specified in addr.

Task-1. Write the client-server program to send messages via sockets (C language)



Task-2. Using this program send message through every students



help: https://www.geeksforgeeks.org/socket-programming-cc/