Socket Programming in C

What is a Socket?

- It is an IPC (inter process communication) mechanism to pass data between processes
  - Processes can be in the same host (UNIX domain sockets)
  - Processes can be in different hosts (Internet domain sockets)

Communication domain

- Defines the addressing format
- UNIX domain: within a host - addressing uses the file system
- Internet domain: across hosts - uses the Internet addresses + port numbers

Domain	Communication performed	Communication between applications	Address format	Address structure
AF_UNIX	within kernel	on same host	pathname	*sockaddr_un*
AF_INET	via IPv4	on hosts connected via an IPv4 network	32-bit IPv4 address + 16-bit port number	*sockaddr_in*
AF_INET6	via IPv6	on hosts connected via an IPv6 network	128-bit IPv6 address + 16-bit port number	*sockaddr_in6*

Socket Types

- SOCK_STREAM: Stream sockets - reliable, bidirectional, byte-stream communication channel
- SOCK_DGRAM: Datagram sockets - information exchanged as messages, message boundaries preserved, not reliable, messages can arrive out-of-order

Property	Socket type	
	Stream	Datagram
Reliable delivery?	Y	N
Message boundaries preserved?	N	Y
Connection-oriented?	Y	N

Socket System Calls

- **socket()** - creates a new socket
- **bind()** - binds a socket to an address - server uses this to bind socket to well known address
- **listen()** - allows a stream socket to accept incoming connections
- **accept()** - accepts from a peer application on listening stream socket
- **connect()** - establishes a connection with another socket
- **read() & write()** to data I/O

Creating a Socket

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

int socket(int domain, int type, int protocol);
```

Returns file descriptor on success, or −1 on error

- domain - communication domain
- type - socket type
- protocol is 0

Binding a Socket to an Address

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

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

Returns 0 on success, or −1 on error

- sockfd - file descriptor obtained from a previous socket() call
- addr - pointer to a structure specifying the address (depends on the socket domain)
- addrlen - specifies the length of addr

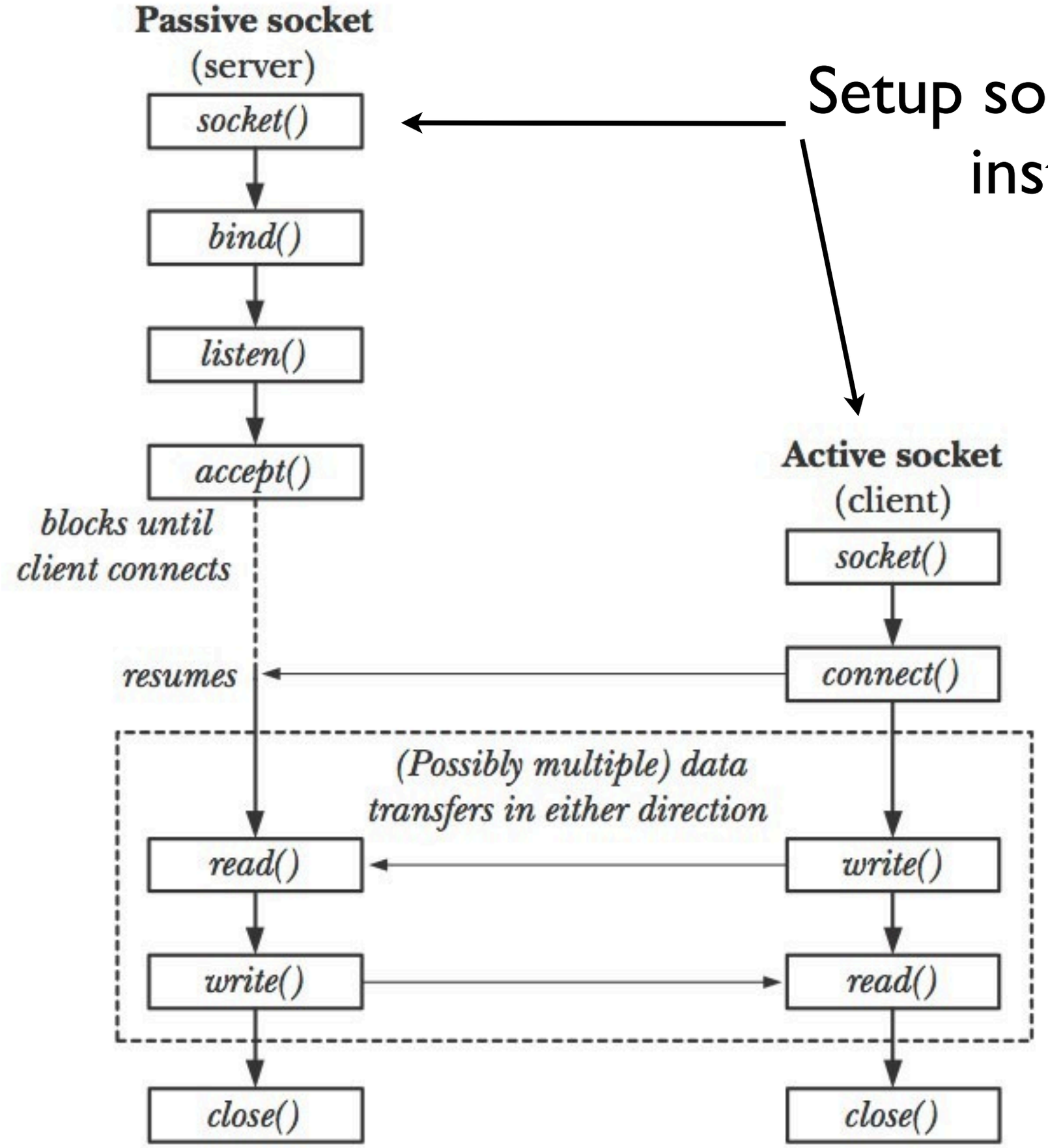
Generic Socket Address Structures

- Sockets use different addresses depending on the domain
- Family - denotes the communication domain
- Structure needs to hold all possible address formats

```
struct sockaddr {
    sa_family_t sa_family;          /* Address family (AF_* constant) */
    char        sa_data[14];        /* Socket address (size varies
                                       according to socket domain) */
};
```

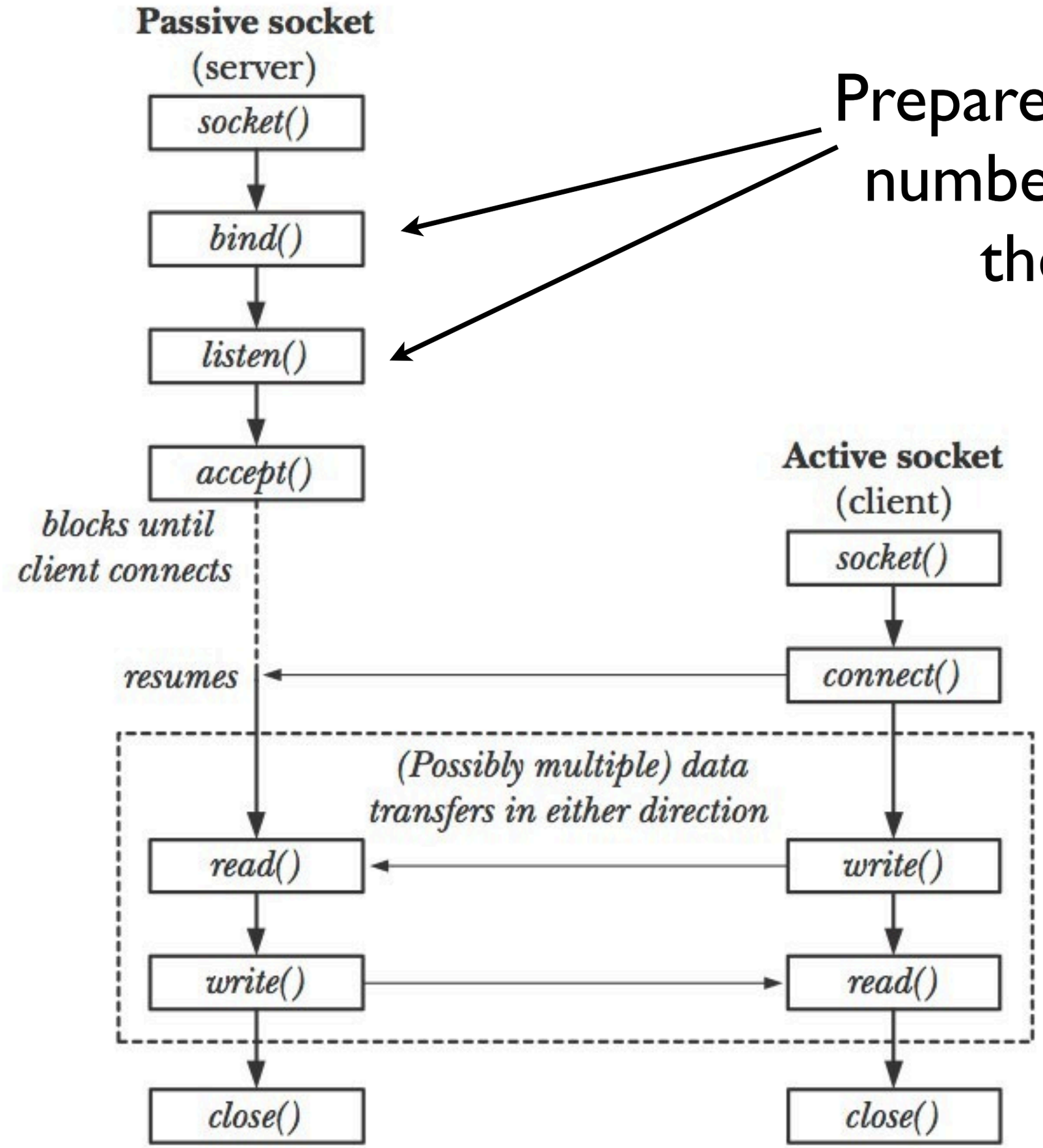
Telephone Analogy

Setup socket - both sides
install phones



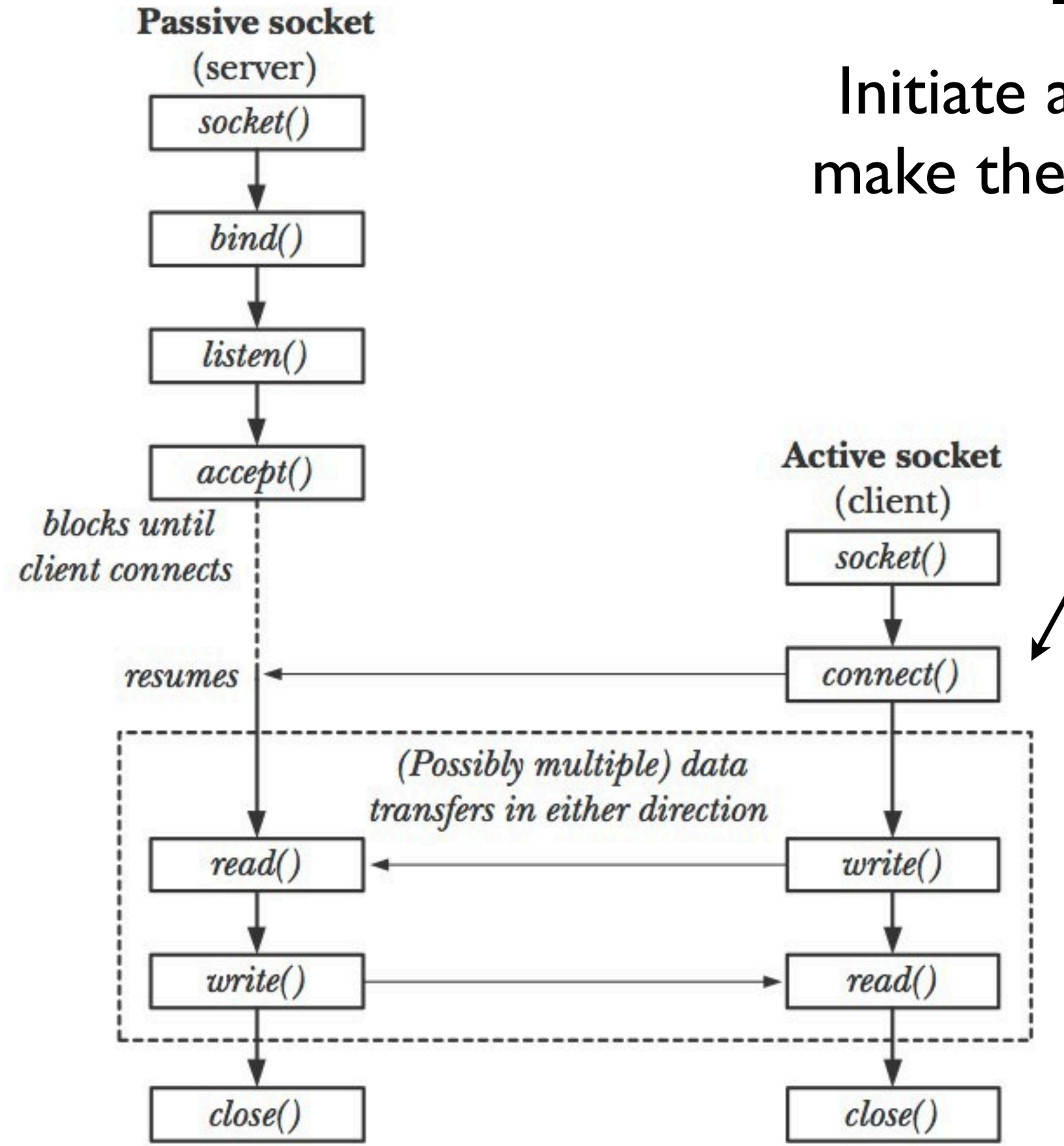
Telephone Analogy

Prepare server end - get number and power up the telephone



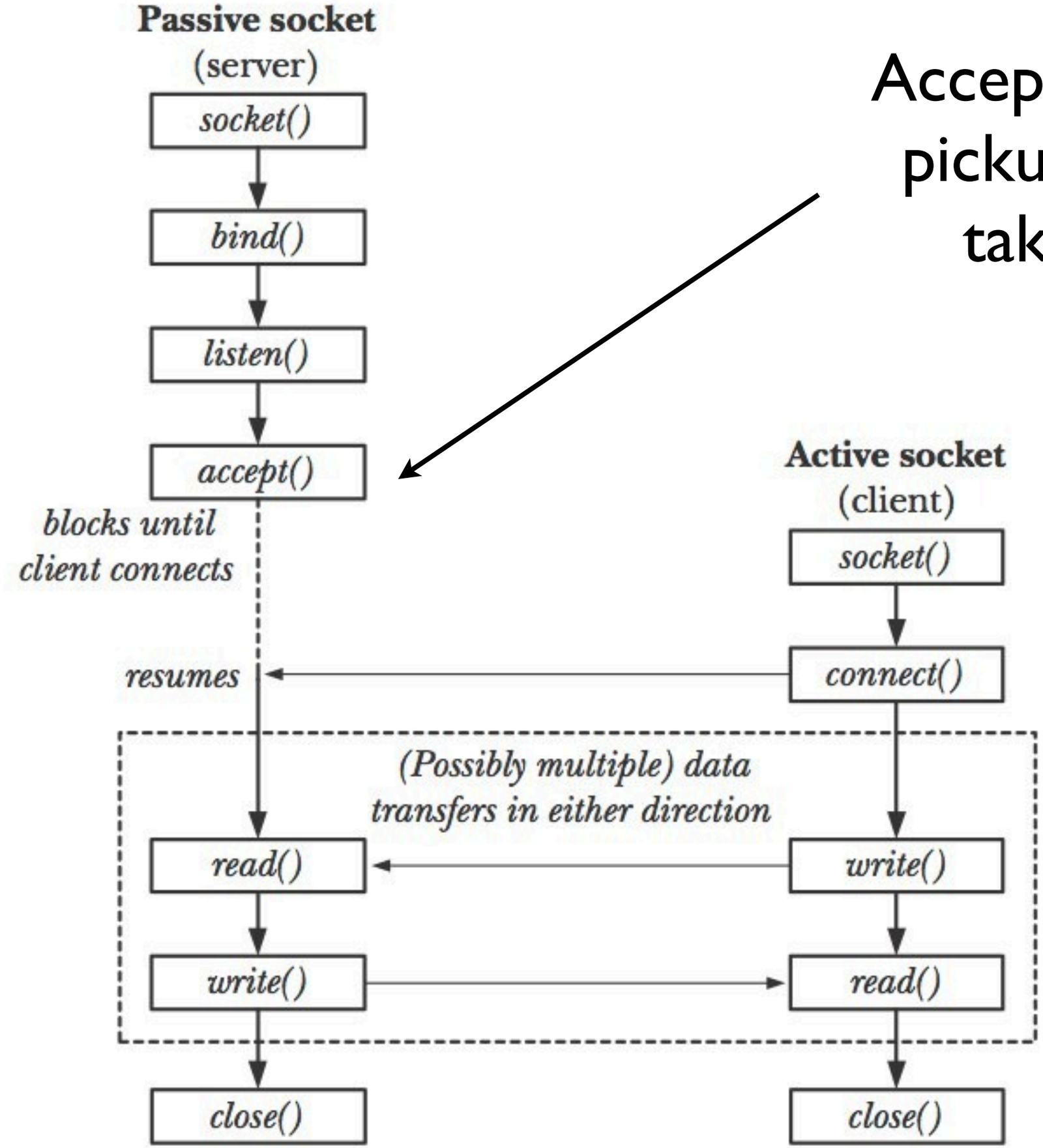
Telephone Analogy

Initiate a connection - make the telephone call



Telephone Analogy

Accept the connection - pickup the receiver to take incoming call



Actual conversation

Active versus Passive Sockets

- Active sockets use connect() call to establish connection to a passive socket - active open
- Passive sockets listen for incoming connections - passive open

Listening for Incoming Connections

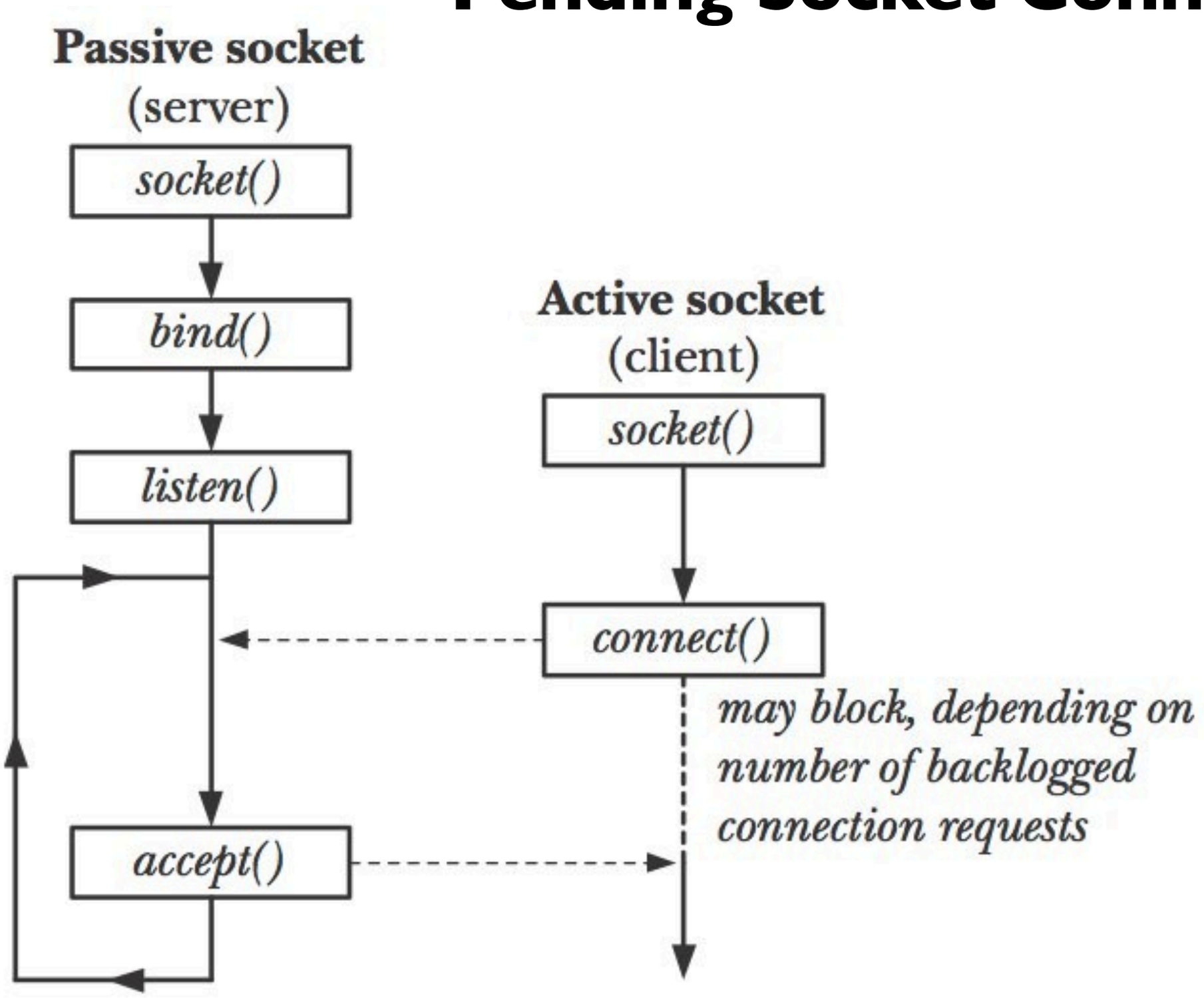
```
#include <sys/socket.h>

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

Returns 0 on success, or −1 on error

- listen() marks the stream socket denoted by sockfd as passive
- backlog specifies number of pending connections

Pending Socket Connection



Accepting a Connection

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

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

Returns file descriptor on success, or −1 on error

- accept() creates a **new socket** that is connected to the peer that performed the connect()
- listening socket remains open and is available to accept further connections

Connecting to a Peer Socket

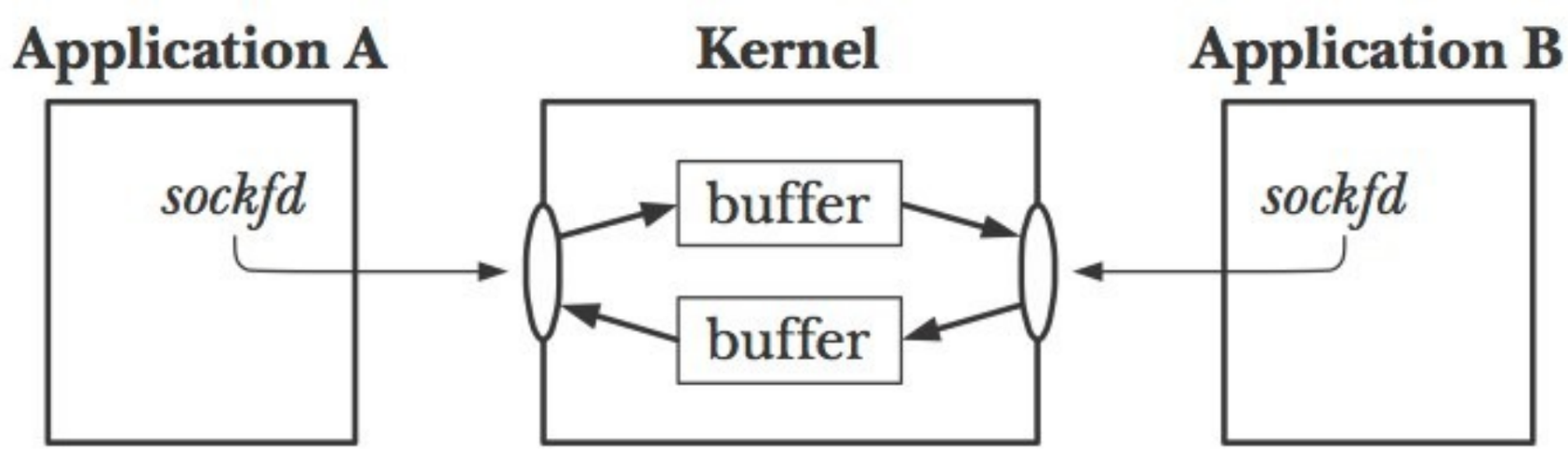
```
#include <sys/socket.h>

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

Returns 0 on success, or −1 on error

- Connect system call connects the active socket to the passive socket that is listening
- Addr and addrlen are specified the same way they are specified in bind()

I/O on Stream Sockets



- Connected sockets provide bidirectional channel
- Socket may be closed by close() or because the application has terminated

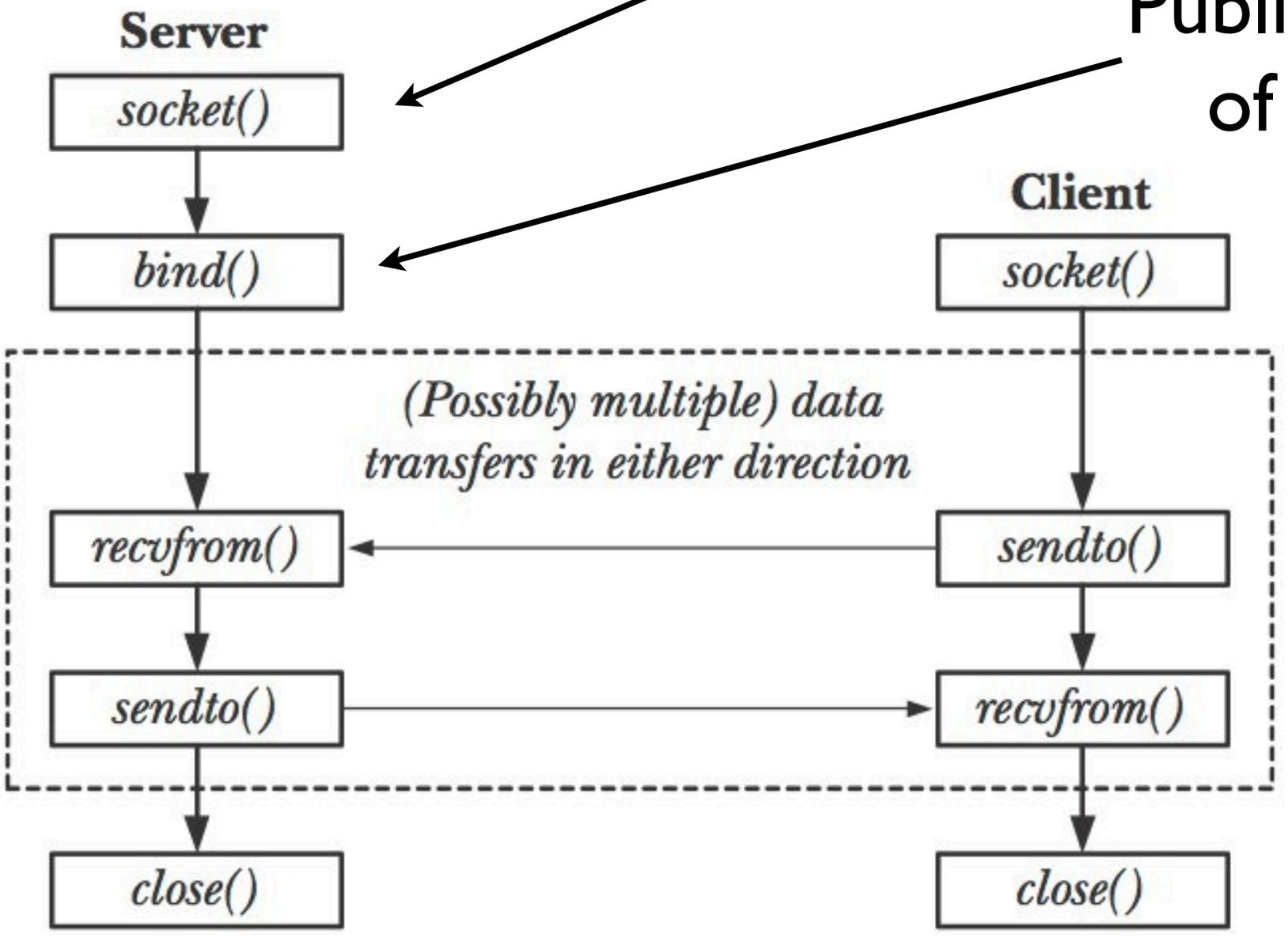
Datagram Sockets

- Explained by analogy with the postal system
- Note - stream sockets were analogous to the telephone system!

Postal Analogy

Setup a mailbox

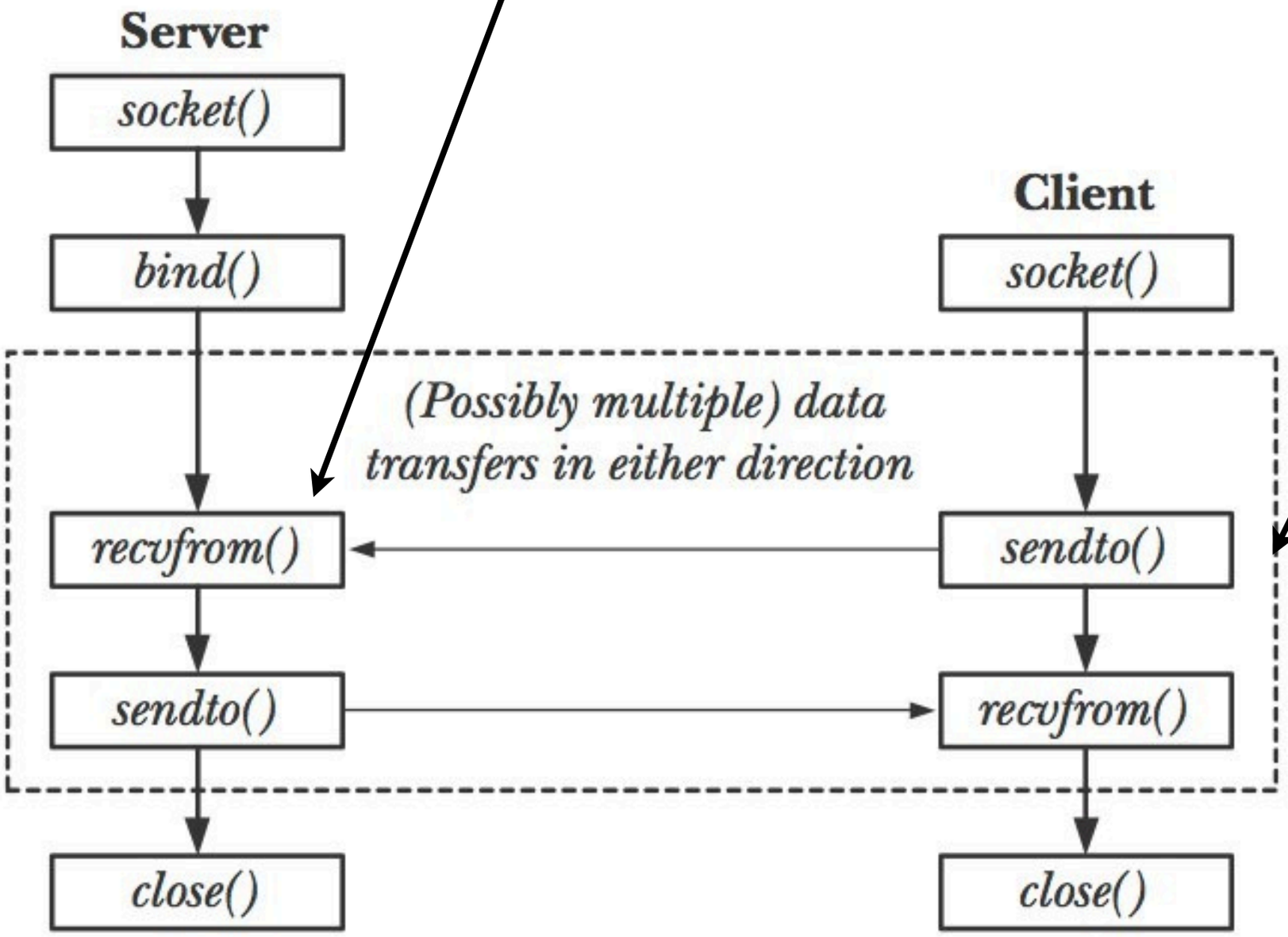
Publish the address of the mailbox



Postal Analogy

Receive the message and reply to the address found in the message

Sender sends a message to the server at its known address



Exchanging Datagrams

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

ssize_t recvfrom(int sockfd, void *buffer, size_t length, int flags,
                 struct sockaddr *src_addr, socklen_t *addrlen);
```

Returns number of bytes received, 0 on EOF, or −1 on error

```
ssize_t sendto(int sockfd, const void *buffer, size_t length, int flags,
               const struct sockaddr *dest_addr, socklen_t addrlen);
```

Returns number of bytes sent, or −1 on error