Course Announcement

- 期末專案分組 (https://goo.gl/Hpmh8o)
  - Due : 11/10 23:55
  - 最多**3**人一組

- 期末考線上課程：(https://goo.gl/WVcJVy)

Lab3

Socket Programming

Socket Introduction

- First introduced in BSD4.1 UNIX, 1981
- A network socket is an endpoint of a connection across a computer network
- An interface which application's process can send and receive messages to/from another application

Basic Concept

Byte Ordering

- Big-Endian
  - Network Byte Ordering
  - The most significant byte (MSB) value is at the lowest address.

- Little-Endian
  - Host Byte Order (ex: Intel x86)
  - The least significant byte (LSB) value is at the lowest address.

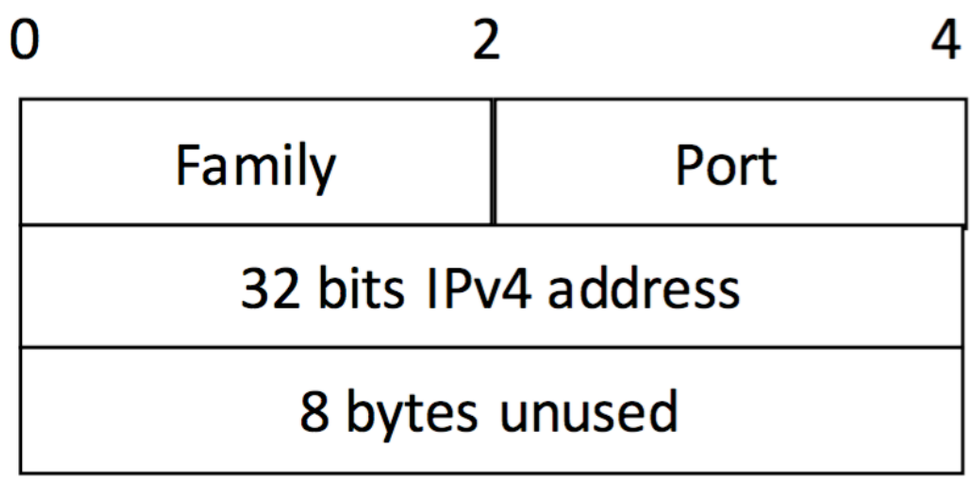
Address	L					H
Big Endian	…	0x12	0x34	0x56	0x78	…
Little Endian	…	0x78	0x56	0x34	0x12	…

Byte Ordering in Socket Programming

```c
#include <netinet/in.h>

// host to network short (2 bytes)
short int htons(short int hostShort);

// host to network long (4 bytes)
long int htonl(long int hostLong);

// network to host short
short int ntohs(short int netShort);

// network to host long
long int ntohl(long int netLong);
```

Specifying Address

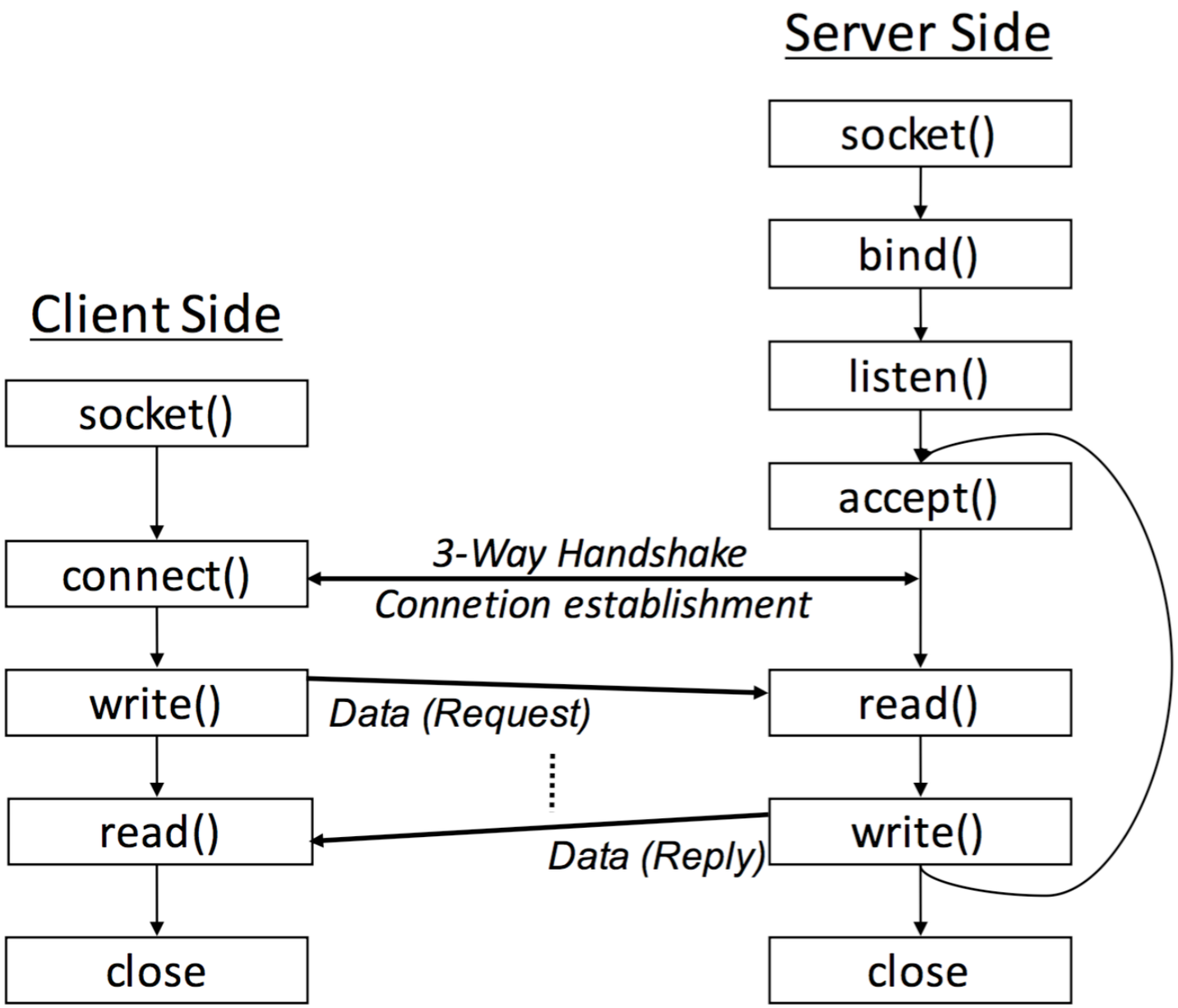
```c
// IPv4 AF_INET sockets
struct sockaddr_in {
     unsigned short  sin_family;  // address family, eg: AF_INET
     unsigned short sin_port;     // address port, eg: htons(5566)
     struct  in_addr sin_addr;    // see struct in_addr, below
     char    sin_zero[8];         // not used
};
struct in_addr {
    unsigned long s_addr;         // internet address, eg: htonl(INADDR_ANY)
};
```



- 0-1023: well-known ports
- 1024-49151: registered ports
- 49152-65535: dynamic ports

Start Socket Programming

TCP Socket Workflow



TCP Socket Workflow in C

- **Server action**
  - Create a socket (socket())
  - Bind to port number (bind())
  - Listen on the socket (listen())
  - Accept client connections (accept())
  - Start Communicate(read()/write())
  - Terminate(close())

- **Client action**
  - Create a socket (socket())
  - Connect to a given port on the server address (connect())
  - Start Communicate(read()/write())
  - Terminate(close())

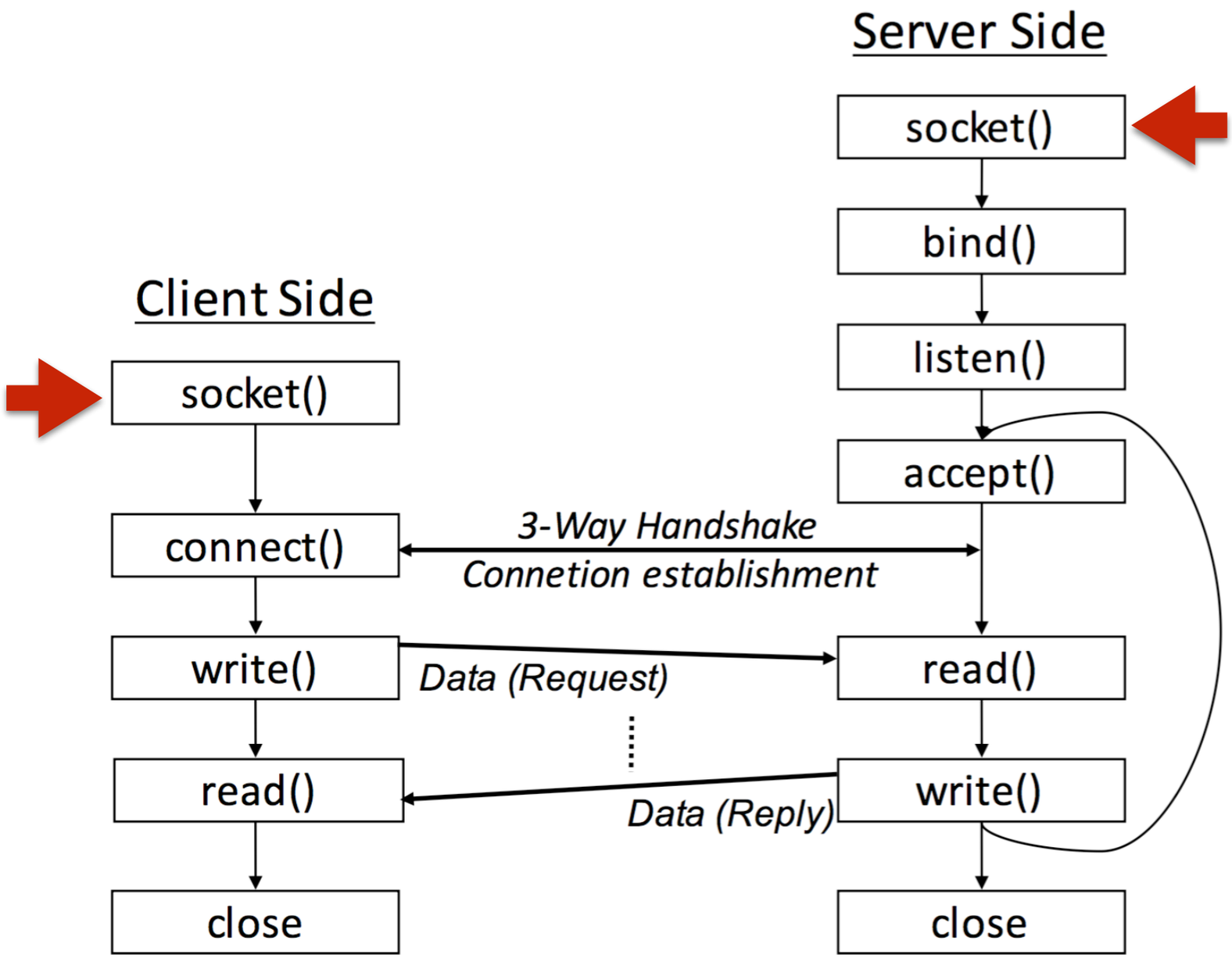
Socket API

Socket API: socket()

Family	Description
AF_INET	IPv4
AF_INET6	IPv6
AF_LOCAL	Unix domain protocols ~ IPC
AF_ROUTE	Routing sockets ~ appls and kernel
AF_KEY	Key socket

Type	Description
SOCK_STREAM	stream socket (TCP)
SOCK_DGRAM	datagram socket (UDP)
SOCK_RAW	raw socket
SOCK_PACKET	datalink (Linux)

TCP Socket Workflow



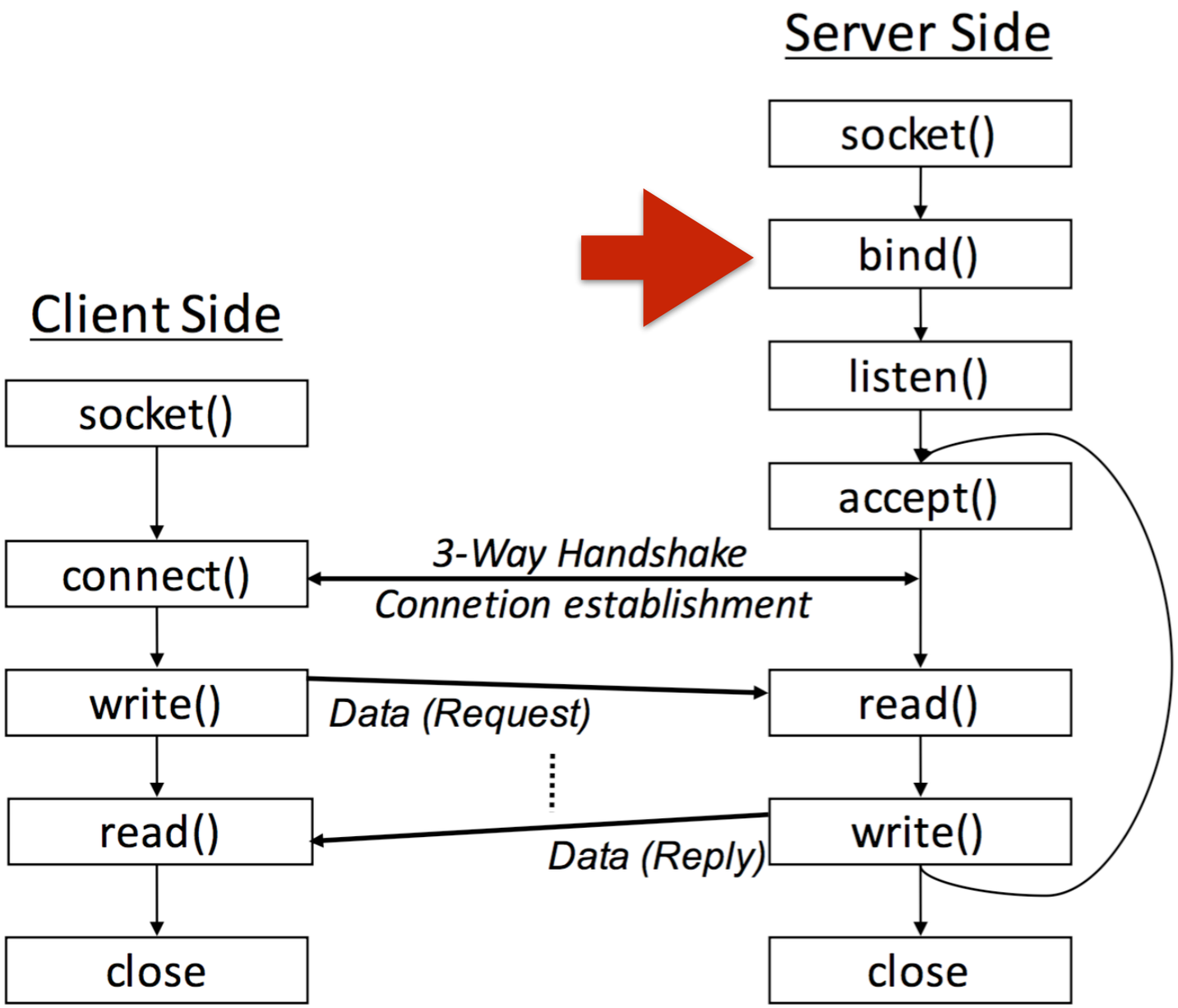
Socket API: socket()

- Creates a TCP or UDP socket
- Return :
  - Integer > 0 (the descriptor of the new socket), if success
  - -1, if error occurs

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

int svr_fd;  // socket file descriptor, return by `socket()`

/* 1) Create the socket, use `socket()` */
svr_fd = socket(AF_INET, SOCK_STREAM, 0);
if (svr_fd < 0) {
  perror("Create socket failed.");
  exit(1);
}
```

Server Side

TCP Socket Workflow



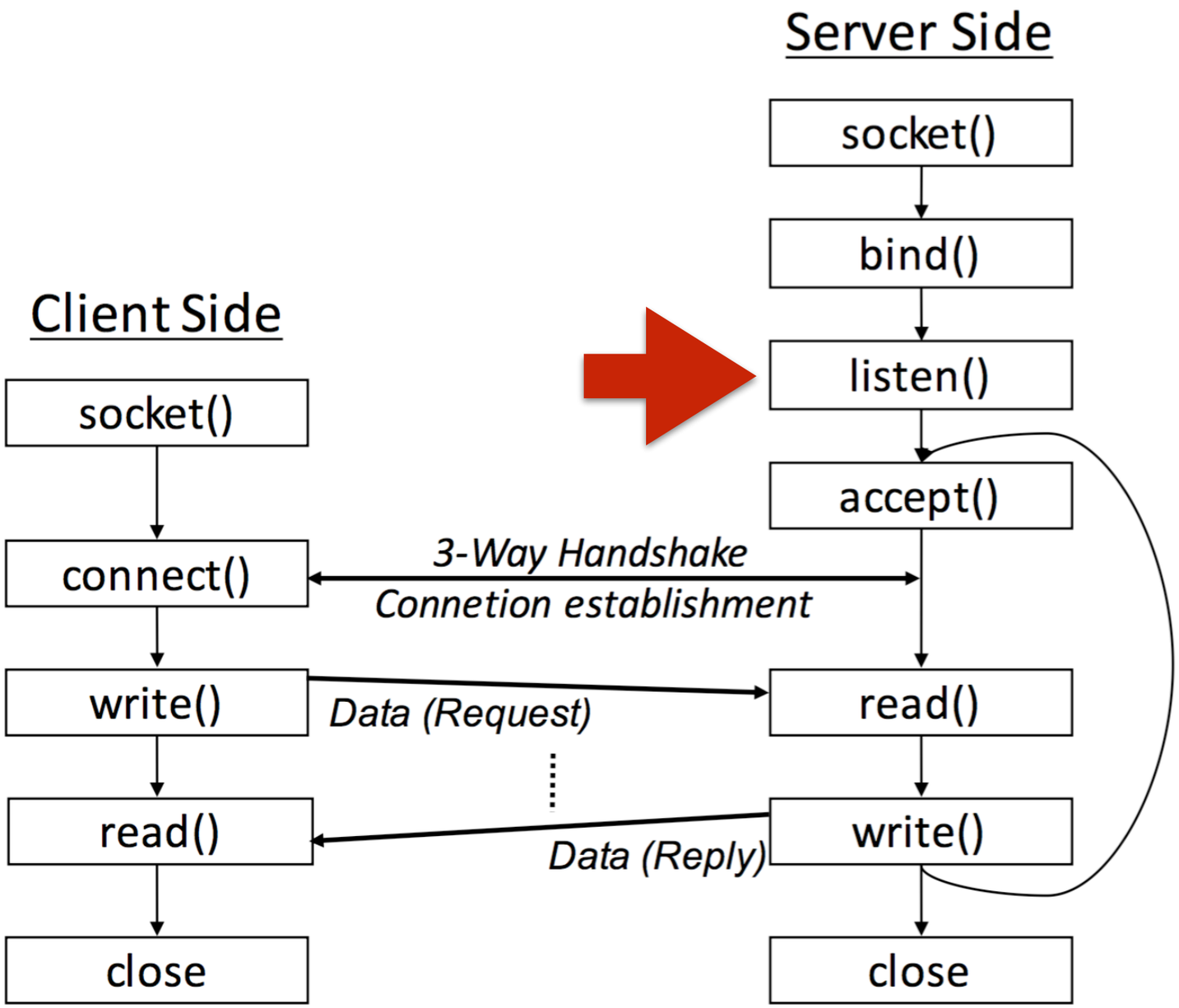
Socket API: bind()

- Bind the given descriptor with the given Internet address and port

- Returns: 0 on success; -1, on failure

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

int svr_fd;                    // socket file descriptor, return by `socket()`
struct sockaddr_in svr_addr;  // address of server, used by `bind()`

/* 1) ... */
/* prepare sockaddr_in */
bzero(&svr_addr, sizeof(svr_addr));
svr_addr.sin_family = AF_INET;
svr_addr.sin_addr.s_addr = htonl(INADDR_ANY);
svr_addr.sin_port = htons(PORT);

/* 2) Bind the socket to port, with prepared sockaddr_in structure */
if (bind(svr_fd, (struct sockaddr*)&svr_addr , sizeof(svr_addr)) < 0) {
  perror("Bind socket failed.");
  exit(1);
}
```

TCP Socket Workflow



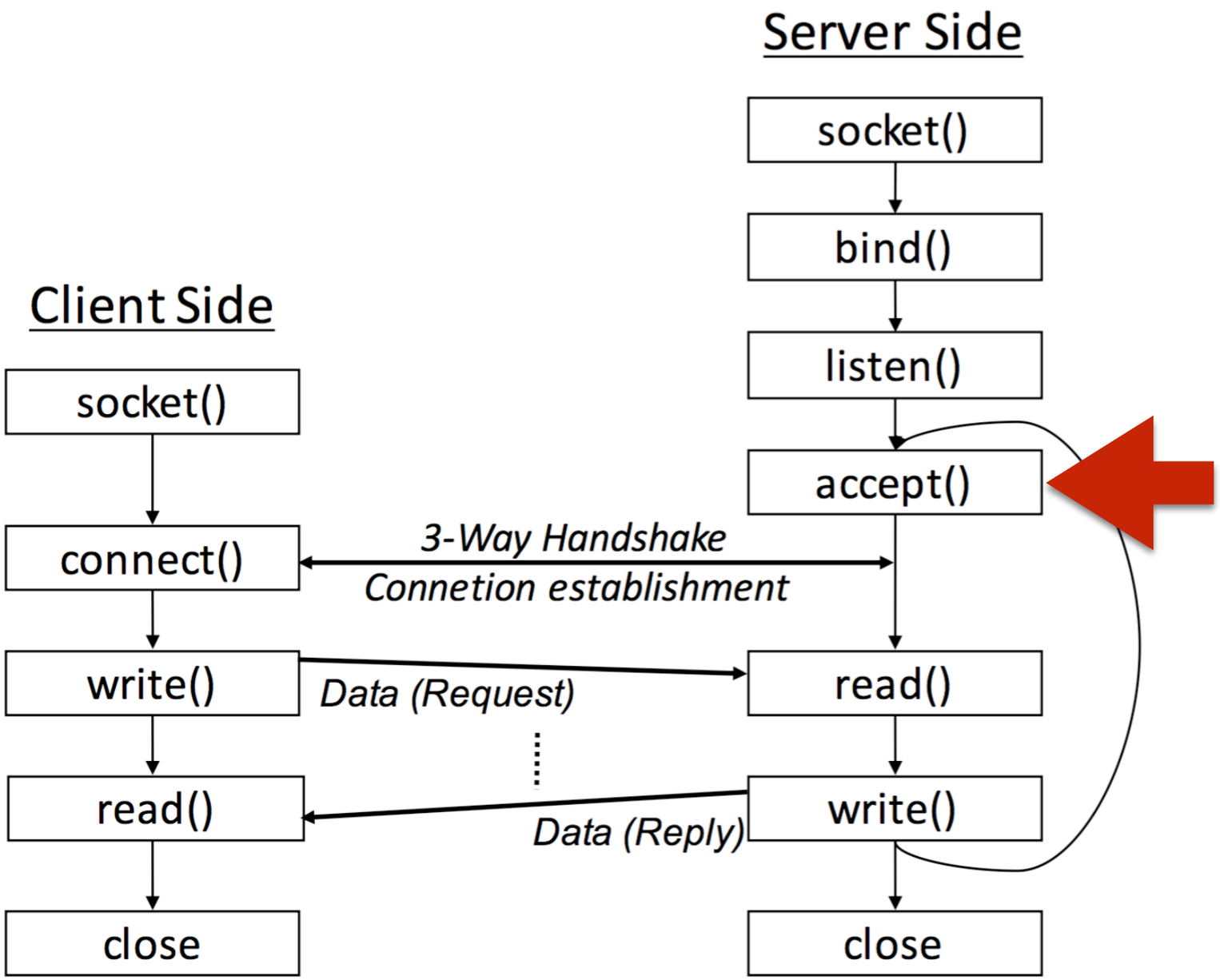
Socket API: listen()

- Listen for connections.The incoming connection requests will be handled and then queued for acceptance by the program.

- Returns: 0 on success; -1, on failure

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

int svr_fd;  // socket file descriptor, return by `socket()`

/* 1), 2) ... */
/* 3) Listen on socket */
if (listen(svr_fd, MAX_CONNECTION) < 0) {
  perror("Listen socket failed.");
  exit(1);
}
```

TCP Socket Workflow



Socket API: accept()

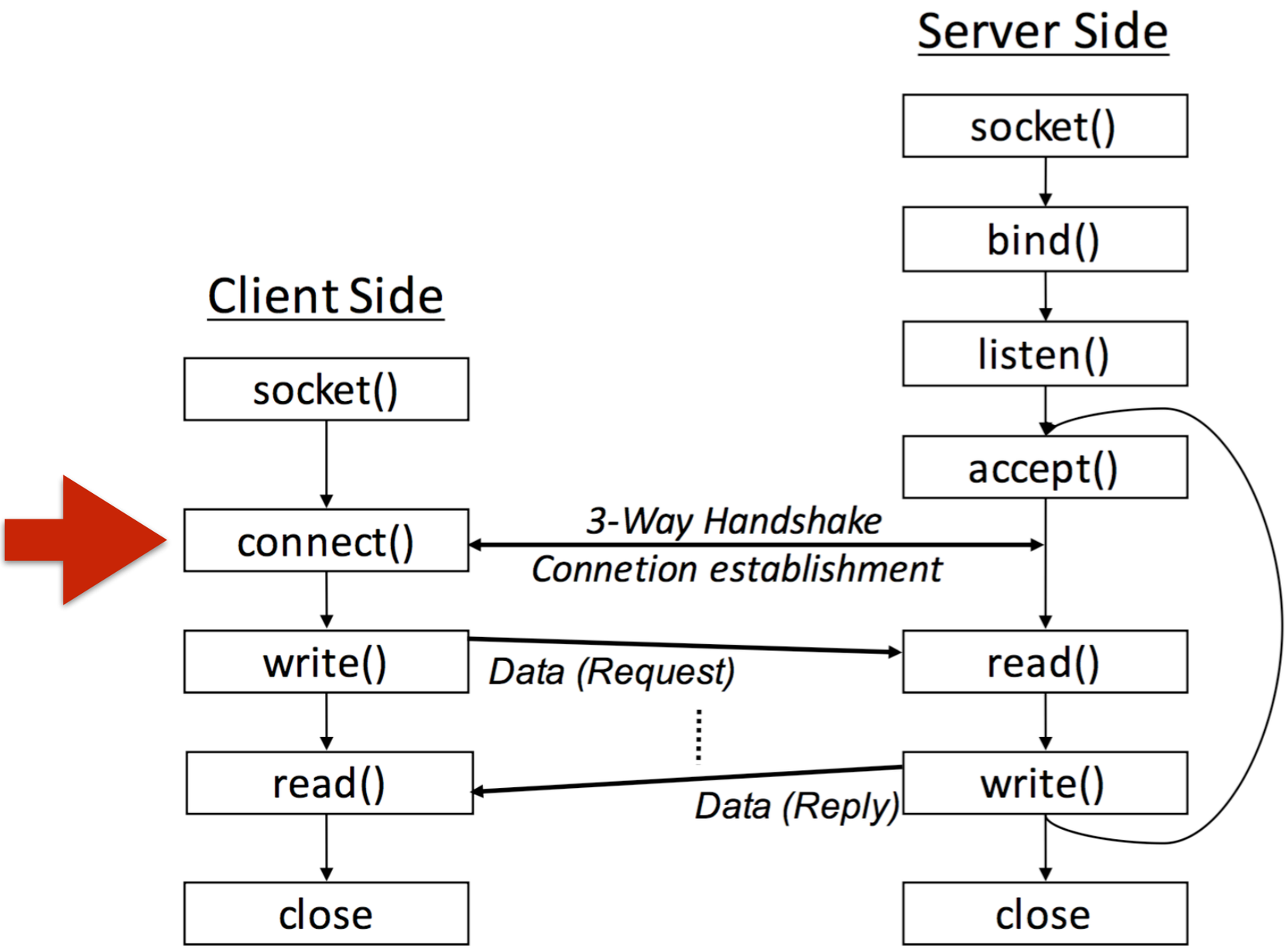
- Dequeues the next connection on the queue for socket. If the queue is empty, accept() blocks until a connection request arrives

- Returns:
  - Integer > 0 (the newly connected socket descriptor), if success
  - -1, if error occurs

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

int svr_fd;          // socket file descriptor, return by `socket()`
socklen_t addr_len; // size of address, used by `accept()`

/* 1), 2), 3) ... */
/* 4) Accept client connections */
addr_len = sizeof(struct sockaddr_in);
cli_fd = accept(svr_fd, (struct sockaddr*)&cli_addr, (socklen_t*)&addr_len);

if (cli_fd < 0) {
  perror("Accept failed");
  exit(1);
}
```

Client Side

TCP Socket Workflow



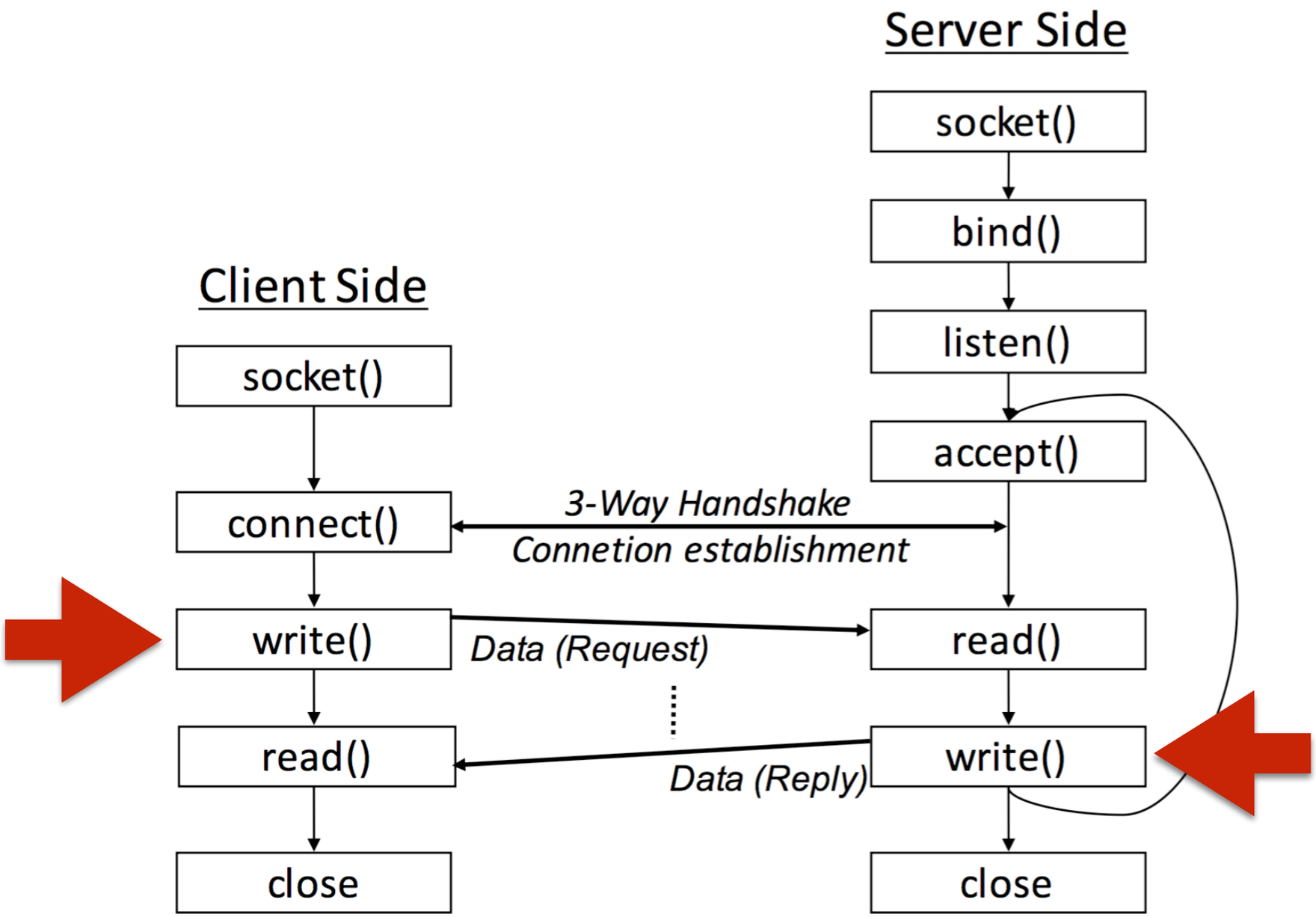
Socket API: connect()

- Establishes a connection between the given socket and the remote socket associated with the foreign address

- Returns: 0 on success; -1, on failure

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

int cli_fd;                     // descriptor of client, used by `socket()`
struct sockaddr_in svr_addr;  // address of server, used by `connect()`

/* 1) ... */
/* prepare sockaddr_in structure */
bzero(&svr_addr, sizeof(svr_addr));
svr_addr.sin_family = AF_INET;
svr_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
svr_addr.sin_port = htons(PORT);

/* 2) Connect to server with prepared sockaddr_in structure */
if (connect(cli_fd, (struct sockaddr *)&svr_addr, sizeof(svr_addr)) < 0) {
  perror("Connect failed");
  exit(1);
}
```

TCP Socket Workflow



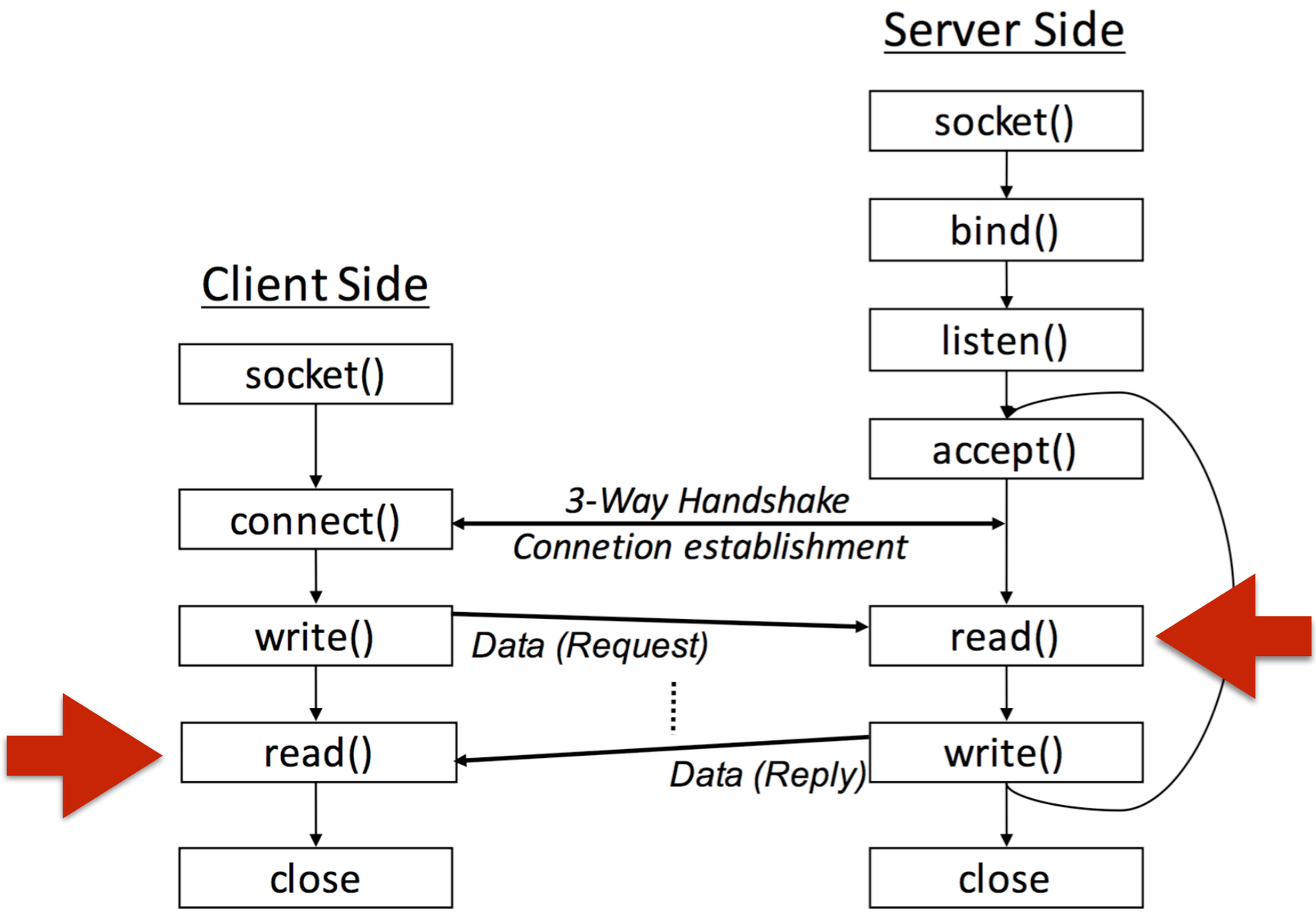
Socket API: write()

- Writes up to count to the server/client referenced by the descriptor.

- Returns:
  - Integer >= 0 (number of bytes written,zero indicates nothing was written)
  - 1, if error occurs

```c
#include <unistd.h>

int cli_fd;          // descriptor of incomming client
int write_bytes;     // number of write byte
char buf[MAX_SIZE]; // buffer to store msg

write_bytes = write(cli_fd, buf, strlen(buf));
if(write_bytes < 0) {
  perror("Write Failed");
  exit(1);
}
```

TCP Socket Workflow

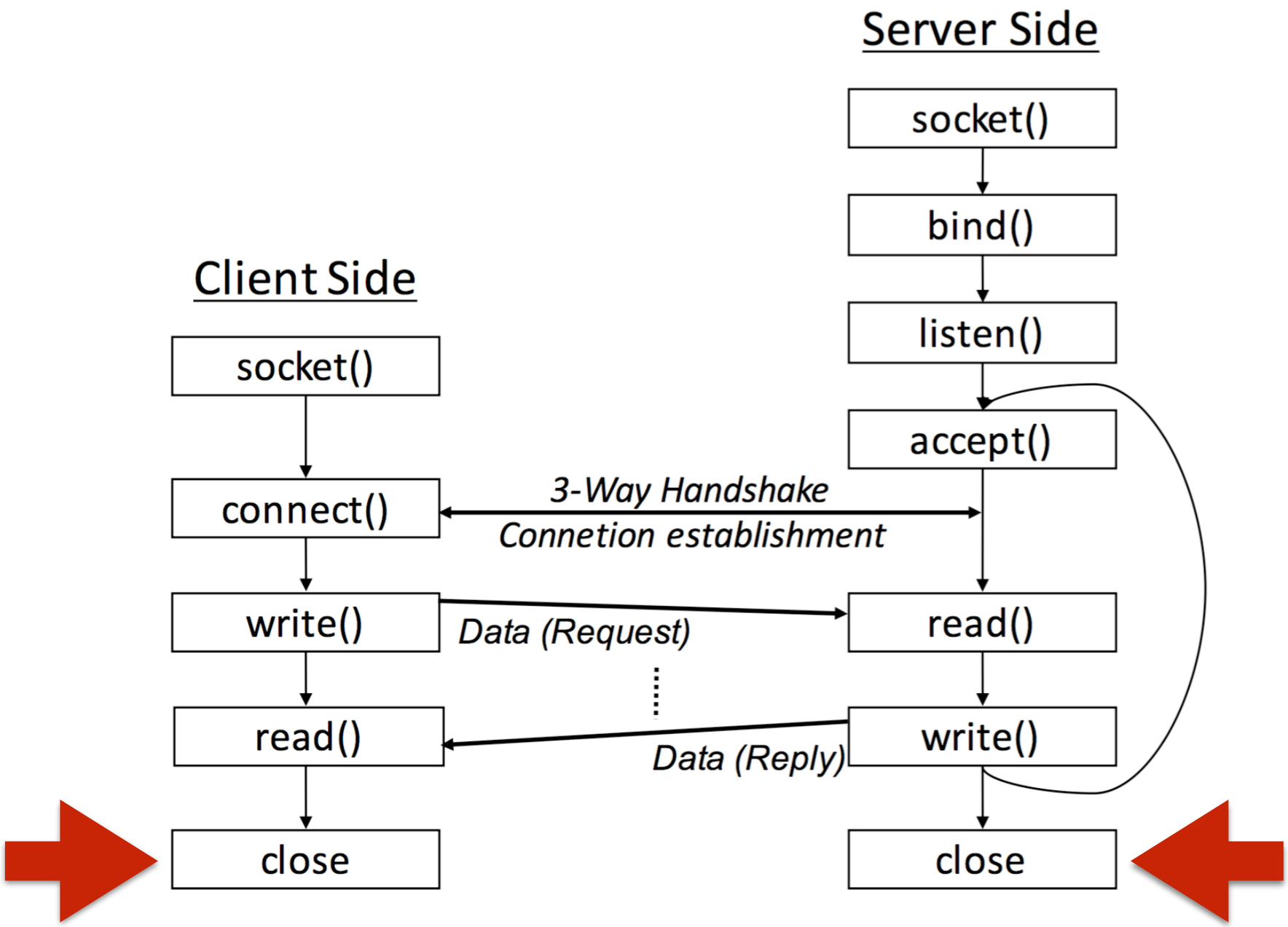


Socket API: read()

- Read up to count bytes from descriptor into the buffer starting at buf
- Returns:
  - Integer>=0(number of bytes read,zero indicates end of file)
  - 1, if error occurs

```c
#include <unistd.h>

int cli_fd;          // descriptor of incomming client
int read_bytes;      // number of read byte
char buf[MAX_SIZE]; // buffer to store msg

read_bytes = read(cli_fd, buf, sizeof(buf));
if (read_bytes < 0) {
  perror("Read failed");
  exit(1);
}
buf[read_bytes] = '\0';
```

TCP Socket Workflow



Socket API: close()

- Terminate communication socket
- Returns: 0 on success; -1, on failure

```c
#include <unistd.h>

int cli_fd; // descriptor of incomming client

close(cli_fd);
```

Today's Mission

Lab 3-1 Echo Server

- Echo Server
  - A simple server that echo message back to connected client.

- Please modify from example code
  - A server will reply current time to connected client.

How to compile?

- Compile: use “gcc”
  - gcc server.c -o server
  - gcc client.c -o client

- You also can use makefile
  - Ch21.3 (http://linux.vbird.org/linux_basic/0520source_code_and_tarball.php#make)

- Run :
  - ./server
  - ./client

Lab 3-1 Echo Server

- Deadline: **11/10 (Fri.) 23:59**
- Name : 你的學號.zip (server.c / client.c)
- Wrong name **-10** point
- Copy will get **0** point
- Delay will get **0** point