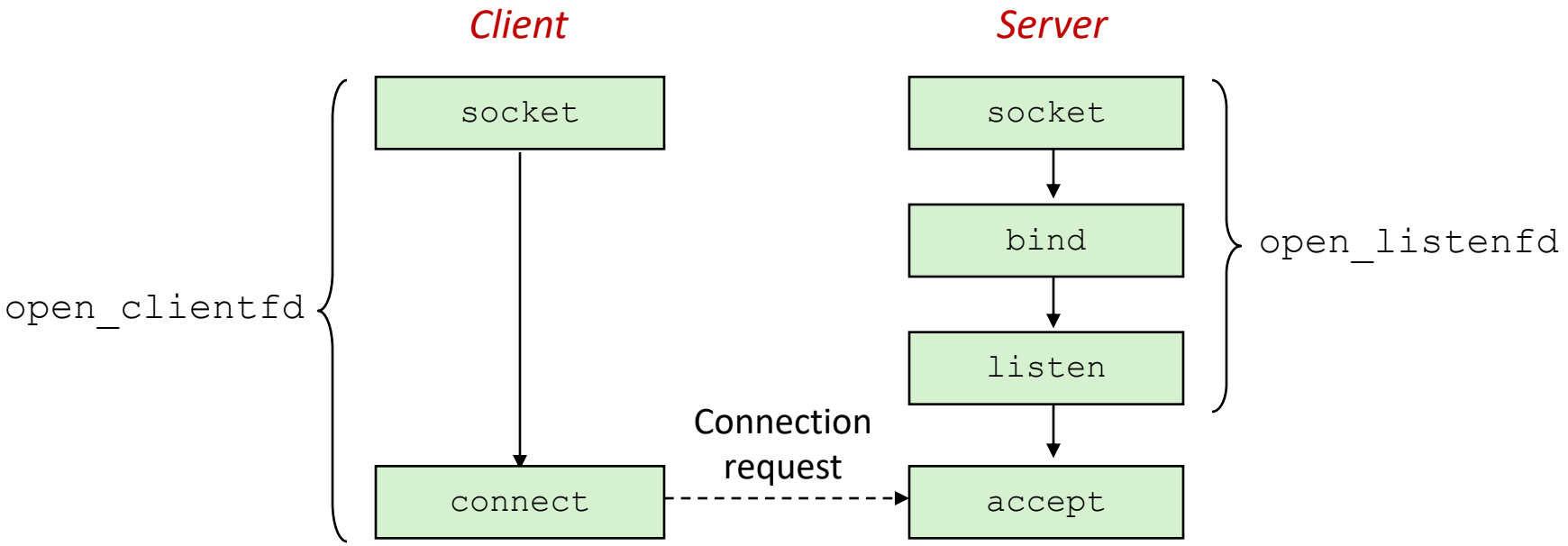
ICS53: Assignment5 discussion

Lab 5: Client/Server

- You will write a networked client/server application which allows a client program to download and upload files from and to a remote server.

Overview of the Sockets Interface



- Office Telephone Analogy for Server
  - Socket: Buy a phone
  - Bind: Tell the local administrator what number you want to use
  - Listen: Plug the phone in
  - Accept: Answer the phone when it rings

Server: **socket**, bind, listen, accept

```
// Creating socket file descriptor
if ((server_fd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP)) == 0)
{
        perror("socket failed");
        exit(EXIT_FAILURE);
}
```

- **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)

Server: **socket**, bind, listen, accept

```
// Forcefully attaching socket to the port 8080
// Set socket options
if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
&opt, sizeof(opt)))
{
        perror("setsockopt");
        exit(EXIT_FAILURE);
}
```

This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as:"address already in use".

- To set options at the socket level, specify the level argument as SOL_SOCKET
- REUSEADDR and REUSEPORT when binding

Server: socket, **bind**, listen, accept

```
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( 8080 );

// Forcefully attaching socket to the port 8080
if (bind(server_fd, (struct sockaddr *)&address, sizeof(address))<0)
{
        perror("bind failed");
        exit(EXIT_FAILURE);
}
```

- 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.

Server: socket, bind, **listen**, accept

```
if (listen(server_fd, 3) < 0)
{
        perror("listen");
        exit(EXIT_FAILURE);
}
```

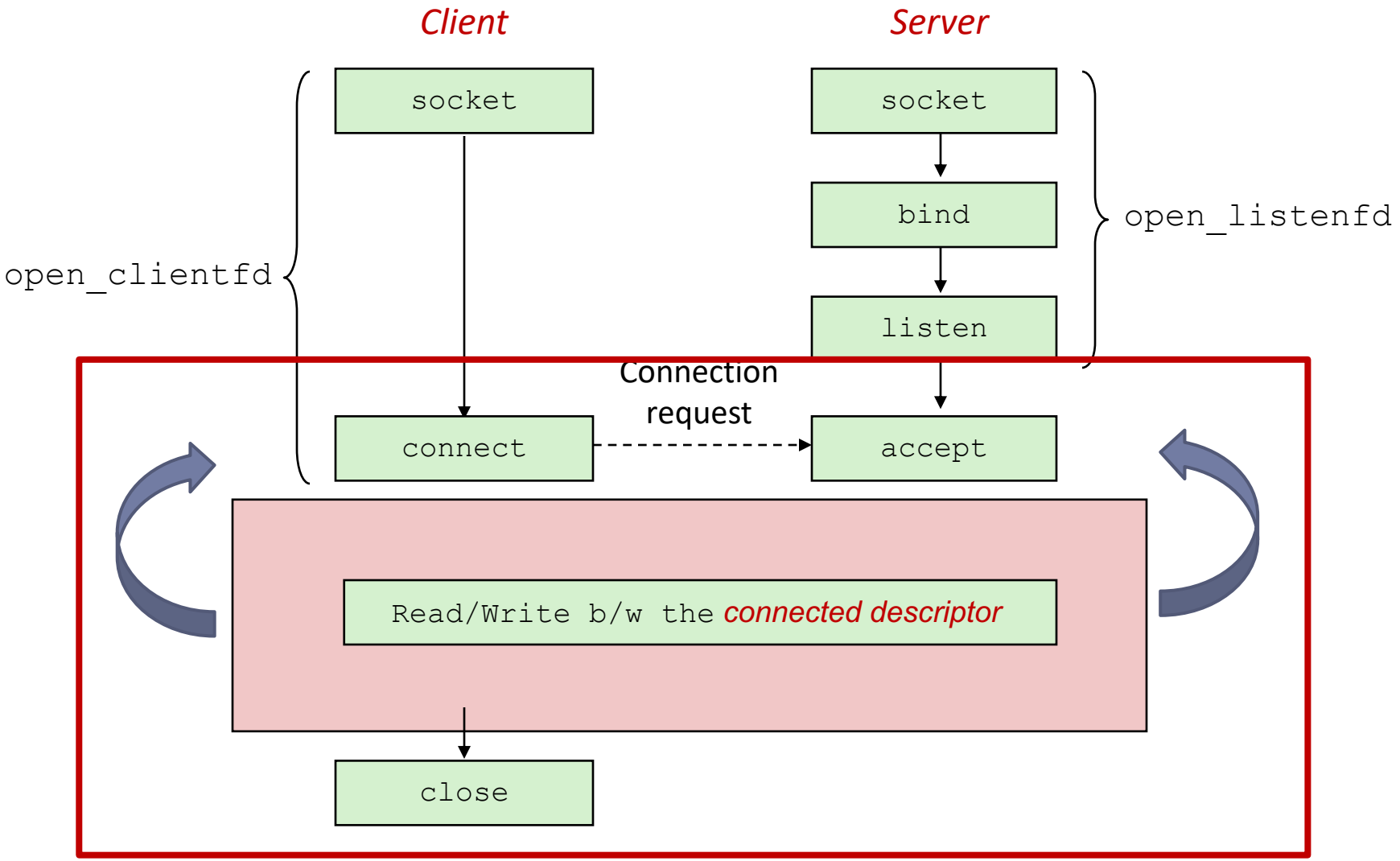
- It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection.
- The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive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

Server: socket, bind, listen, **accept**

```
if ((new_socket = accept(server_fd,
(struct sockaddr *)&address, (socklen_t*)&addrlen))<0)
{
        perror("accept");
        exit(EXIT_FAILURE);
}
```

- 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.

Overview of the Sockets Interface



Server: Main Loop

- The server loops endlessly, waiting for connection requests, then reading input from the client, and respond back to the client.

```
main() {

   /* create and configure the listening socket */

   while(1) {
      /* Accept(): wait for a connection request */
      /* echo(): read and echo input lines from client till EOF */
      /* Close(): close the connection */
   }
}
```

Client: **socket**, connect

```
// Create socket
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
        printf("\n Socket creation error \n");
        return -1;
}
```

Client: socket, **connect**

```
// Connect
struct sockaddr_in serv_addr;
memset(&serv_addr, '0', sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(8080);

if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
{
        printf("\nConnection Failed \n");
        return -1;
}
```

- 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.