Workshop on SL-V @ SCOE, Pune

By
M. R. Penurkar
MITCOE, Pune

Agenda

- Assignment 4
  - Design a distributed application which consists of a server and client using threads.

Requirements

- Distributed System
  - Applications are distributed on minimum two machines.

- Computer Networks
  - TCP/UDP

- Operating systems:
  - System calls for message passing through the underlying network.
  - Threads

Message Passing in Distributed System

- Middleware
  - RPC
  - RMI

- Message Passing Libraries
  - Openmpi
  - MPICH

Message Passing in Distributed System

- The message passing is done by using underlying operating systems.
- Different system calls for different Operating System.
- socket is an example of the system call in the underlying OS in Linux/UNIX/Windows.

What is a Socket ?

- An interface between application process and end-end-transport protocol (UDP or TCP).
    - File descriptor, allows applications to read/write to the network

- Allows to processes on remotely connected computers to talk to each other.

- Socket : - IP Address + Port No.

Options

.Windows – winsock interface

.LINUX/UNIX –socket interface

.Languages – C, C++, Java

Linux Network Design Options

- **TCP**
  - **SOCK_STREAM**
    - Connection oriented, bidirectional
    - Reliable, in-order delivery
- **UDP**
  - **SOCK_DGRAM**
    - Connection-less
    - Unreliable delivery, no guarantee on the order

Socket programming with TCP

- At Server
- Server must a create socket that accepts client’s contact (listening socket).
- First the Server process must be started

- At Client
- Client creates a local TCP socket – specifies IP address, port number of server process.
- Client TCP establishes connection to server TCP (a three way handshake)

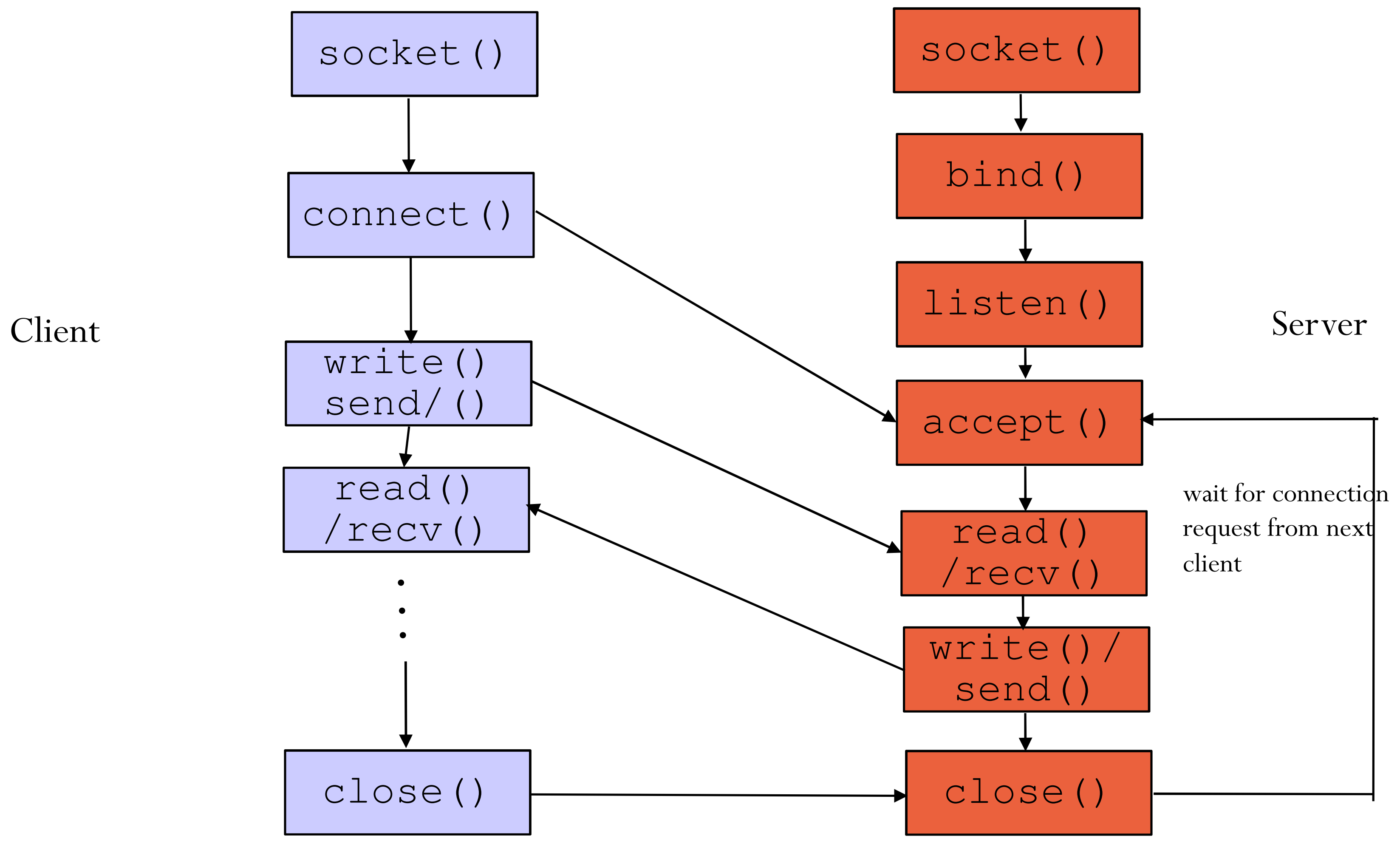
Socket programming with TCP

- At Server
- When client connects to server, server TCP creates new socket (data socket) for server process to communicate with client.
- Allows server to talk with multiple clients
- Source port numbers used to distinguish different clients

Ports

- Ports are used to address processes on a host
- 0-1024 is usually reserved for known service
- Examples:-
  - Web server (Application) -http -80
  - File transfer (Application) –ftp-21

Socket Programming



socket()

```
int sock_id = socket(family, type,
protocol);
```

- family: AF_INET specifies IPv4
- type: SOCK_STREAM, SOCK_DGRAM
- protocol: 0 (IP) (/etc/protocols)

bind()

- **`bind(sock_id,localAddr,AddrLength)`**
  - Server specifies which port and address it will be listening to.
  - sock_id: Listening socket descriptor
  - localAddr: Socket address structure
  - addLength: Length of localAddr

Address Structure

```
struct sockaddr_in {
u_char sin_family;  // family of address
u_short sin_port;  // protocol port num
struct in_addr sin_addr;// IP Addr

};
Defined in netinet/in.h
```

Address Structure

- Declare address structure
  ```
  struct sockaddr_in sockAdd;
  ```

- Set family
  ```
  sockAdd.sin_family = AF_INET;
  ```

- Set IP address

- //listen to any local address
  ```
  sockAdd.sin_addr.s_addr = htonl(INADDR_ANY)
  ```
  ```
  (INADDR_ANY:-0.0.0.0)
  ```

- Set port
  ```
  sockAdd.sin_port = htons(9999);
  ```

listen()

- `int lit = listen(sock_id, queuelength);`
  - lit : -1 if error, 0 otherwise
  - `sock_id`: socket descriptor
  - queuelength: Number of clients that can “wait” for a connection
  - listen is **non-blocking**: returns immediately

accept()

- `int sid_new = accept(sock_id, &clientAddress, &addLength);`
    - `sid_new`: new socket for communication with client
    - `sock_id`: the listening socket
    - clientAddress: struct sockaddr (address of client).
    - addLength: size of client address structure
    - accept is **blocking**: waits for connection before returning

connect()

- **int connect(int** *sockfd*, **const struct sockaddr ****serv_addr*, **socklen_t** *addrlen***);**
  - The **connect**() system call connects the socket referred to by the file descriptor *sockfd* to the address specified by *serv_addr*.
  - The *addrlen* argument specifies the size of *serv_addr*.
  - The format of the address in *serv_addr* is determined by the address space of the socket *sockfd*;

Reading /Writing

- Reading /Writing can be done by read and write system calls respectively.
- Reading /Writing can be done by receive and send system calls respectively.

Design options for Concurrent clients on UNIX/Linux

- Language - C
- Multi –process server
  - Using fork()
  - Creates a new process per client connection
- Multithreaded server
  - Using pthread_create()
  - Creates a new thread per client connection
- Used to keep listening on socket and talking on another socket

pthread_create()

.#include <pthread.h>

. int pthread_create(pthread_t **thread*, const pthread_attr_t **attr*, void *(**start_routine*)(void*), void **arg*);

.pthread_t :- thread id type..

.pthread_attr_t :- attributes of a thread

. *start_routine:- The routine to be run by a thread.*

.*arg:- an argument to the thread start routine.*

Demo

- A Simple client-server echo server.
    - Language :- C
    - OS- Linux (Ubuntu)
    - Thread API- POSIX Threads (pthreads)

Questions ???

Thank You !