CSCI4430 Data Communication and Computer Networks

Socket Programming for UDP
Hints for Assignment 1

HAN Shujie

sjhan@cse.cuhk.edu.hk

Jan. 31, 2019

Outline

- Socket Programming for UDP
  - Introduction
  - Basic Socket Programming for UDP
- Hints for Assignment 1
  - Main structure of Client and Server
  - Send and Receive Data
  - Reusable port
  - List Files
  - Tips

Outline

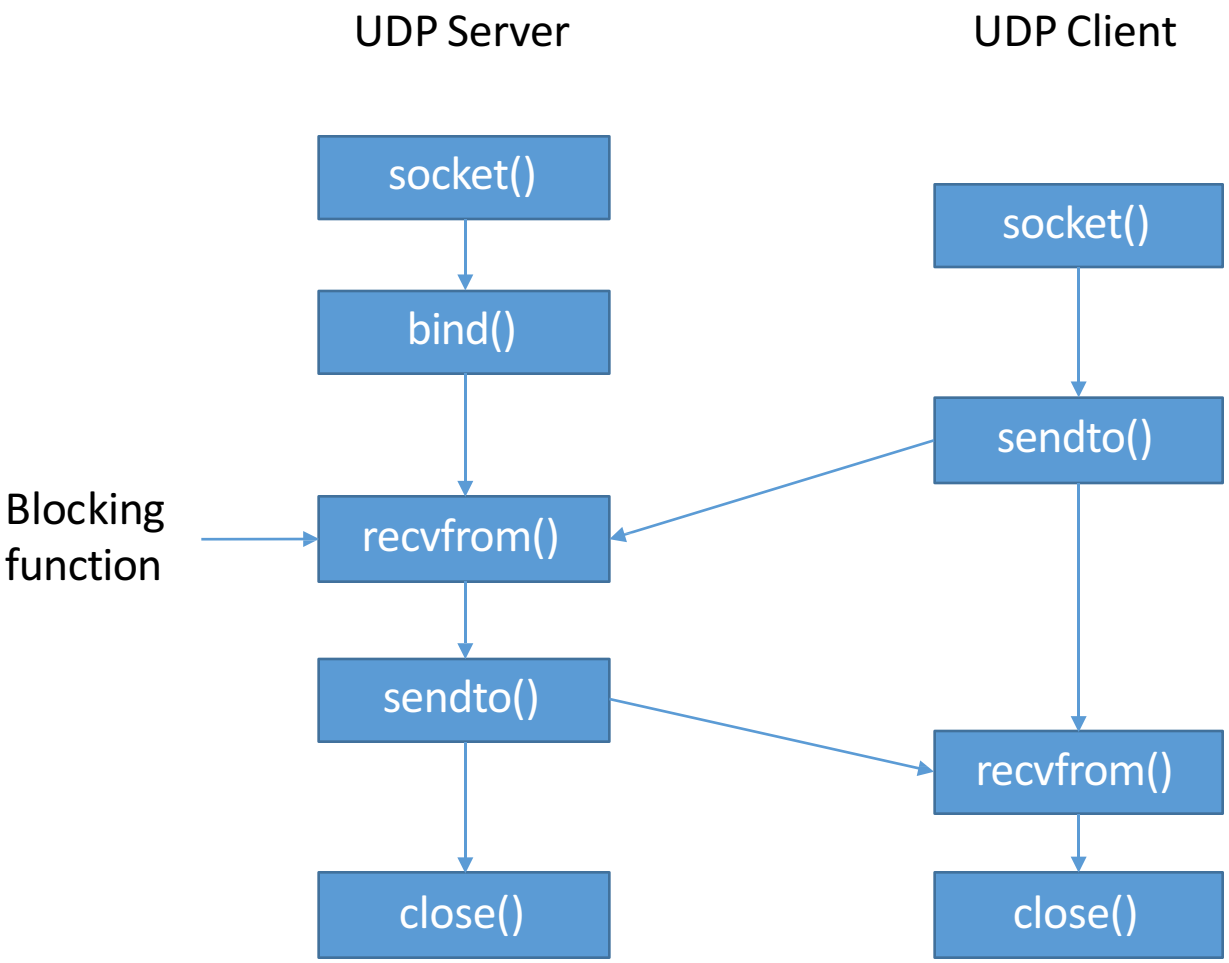
- Socket Programming for UDP
  - Introduction
  - Basic Socket Programming for UDP

- Hints for Assignment 1
  - Main structure of Client and Server
  - Send and Receive data
  - Reusable port
  - List Files
  - Tips

UDP (User Datagram Protocol)

- Another transport layer protocol (recall TCP).
- Difference from TCP:
  - TCP guarantees reliable data transmission by establishing connection between client and server.
  - UDP does not establish connection between server and client. It is a **connection-less** protocol.
    - no handshaking between UDP sender, receiver
    - each UDP segment handled independently of others
  - UDP, thus, does not guarantee the arrival, arrival time and content of the message.

UDP Socket Programming Overview



UDP Socket Programming

- socket()
  - Syntax:
    - int socket(int *family*, int *type*, int *protocol*);
  - To create a UDP socket:
    - int sd = socket(AF_INET, SOCK_DGRAM, 0);

UDP Socket Programming

- bind() (Server)
  - Syntax:
    - int bind(int *sockfd*, const struct sockaddr **addr*, socklen_t *addrlen*);
  - To assign the address to the socket

```
struct sockaddr_in server_addr;
memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(PORT);
socklen_t addrLen = sizeof(server_addr);
if (bind(sd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
    printf("bind error: %s (Errno:%d)\n", strerror(errno), errno);
    exit(0);
}
```

UDP Socket Programming

- sendto() and recvfrom()
  - Syntax:
    - sendto(int *sd*, void* *buf*, int *bufLen*, int *flags*, struct sockaddr* *saddr*, int *addrlen*);
    - recvfrom(int *sd*, void* *buf*, int *bufLen*, int *flags*, struct sockaddr* *saddr*, int* *addrlen*);
  - Example:

```
char* buff=”hello”;
sendto(sd, buff, strlen(buff), 0, (struct sockaddr *)&server_addr, addrLen);
```

UDP Socket Programming

- sendto() and recvfrom()
  - Return value: how many bytes are successfully sent.
  - The return value may not be equal to the argument bufLen.
  - Use a while loop to send and receive data.
- recvfrom()
  - Similar to recv(), recvfrom() is blocking and waits for data from the other side.
  - The struct *sockaddr* saddr* argument
    - After executing recvfrom(), the address of the sender is thus stored in *saddr.
    - We can then send the data using the address *saddr.

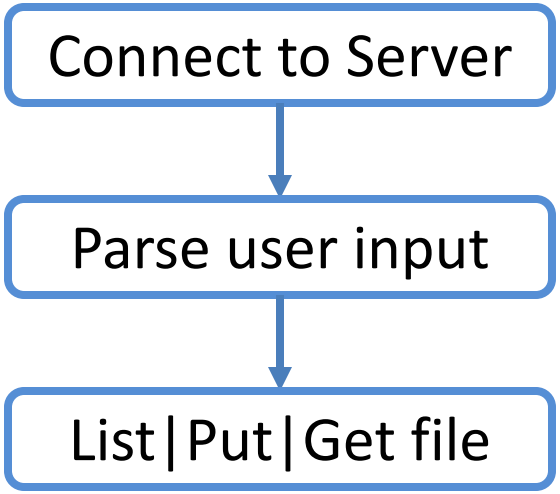
Outline

- Socket Programming for UDP
  - Introduction
  - Basic Socket Programming for UDP

- Hints for Assignment 1
  - Main Structure of Client and Server
  - Send and Receive Data
  - Reusable Port
  - List Files
  - Tips

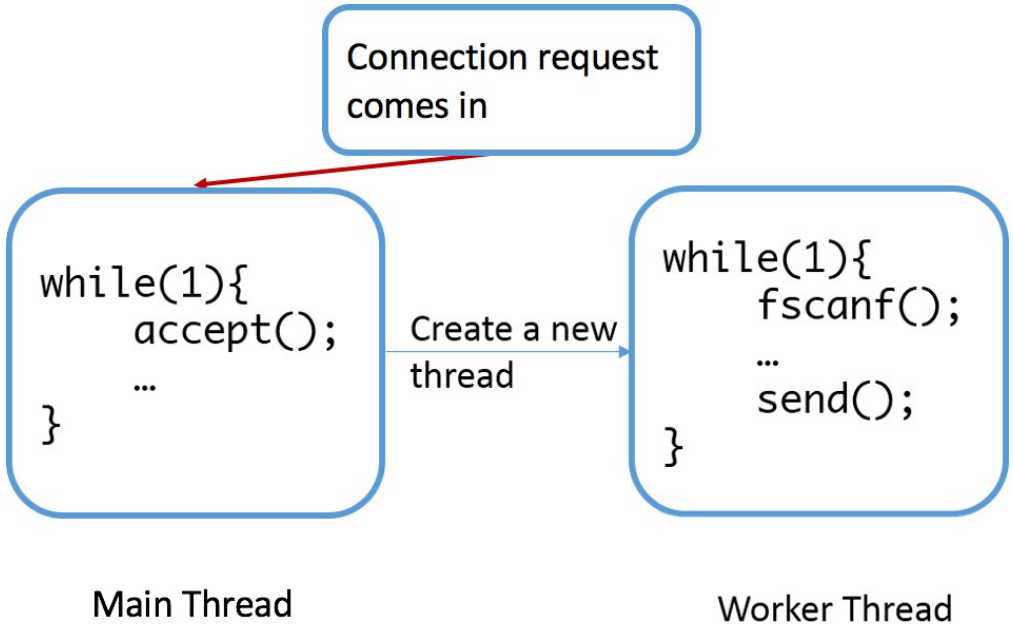
Structure of Client

- A single-thread program is fine for the client side.
- Structure of client:

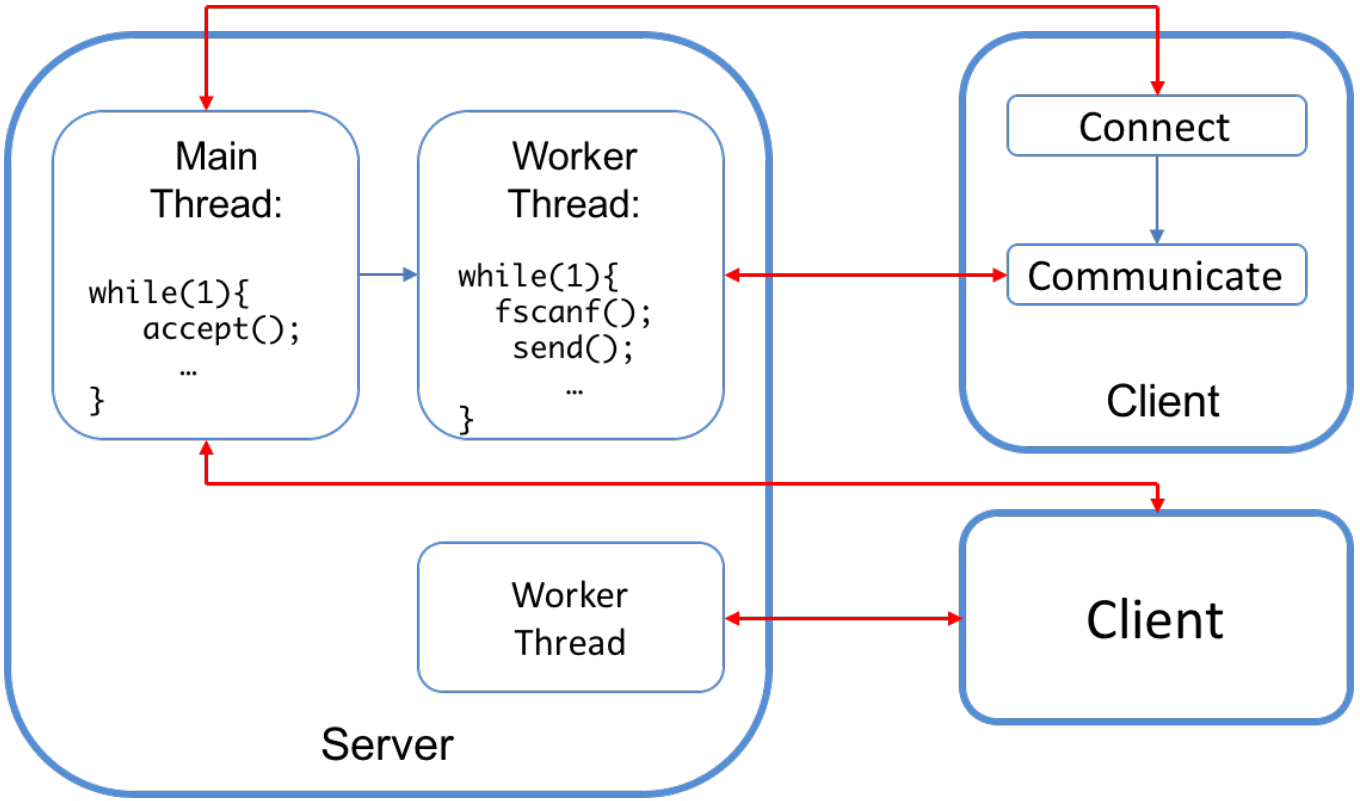


Structure of Server

- Functionalities of server side.
  - Accept connection requests
  - Communicate with client(s)



Whole Picture of The Program



Send and Receive Data

- Write sendn() to ensure complete data transfer

```c
// Send data to ensure complete data transmission
int sendn(int sd, void *buf, int buf_len) {
  int n_left = buf_len;
  int n;
  while (n_left > 0) {
    if ((n = send(sd, buf + (buf_len - n_left), n_left, 0)) < 0) {
      if (errno == EINTR)
        n = 0; // EINTR: interrupt
      else
        return -1;
    } else if (n == 0) {
      return 0;
    }
    n_left -= n;
  }
  return buf_len;
}
```

Send and Receive Data

- Write recvn() to ensure complete data transfer

```c
// receive data
int recvn(int sd, void *buf, int buf_len) {
  int n_left = buf_len;
  int n;
  while (n_left > 0) {
    if ((n = recv(sd, buf + (buf_len - n_left), n_left, 0)) < 0) {
      if (errno == EINTR)
        n = 0; // EINTR: interrupt
      else
        return -1;
    } else if (n == 0) {
      return 0;
    }
    n_left -= n;
  }
  return buf_len;
}
```

Reusable Port

- Reusing server port
  - Generally, if a server crashes, when restart the server and try to bind to the same address/port. You will get an error message “bind: address already in use”
  - To avoid this, you need to make the port reusable

- Adding the following lines after calling socket():

```
long val = 1;
if (setsockopt(server_sd, SOL_SOCKET, SO_REUSEADDR, &val, sizeof(long)) ==
    -1) {
  perror("setsockopt");
  exit(1);
}
```

List Files

- readdir()
  - syntax: *struct dirent *readdir(DIR* dirp);*
  - argument: *dirp* points to the directory that you want to look up.
  - return value: a pointer to *dirent* structure representing the next directory entry. It returns NULL on reaching the end of the directory stream or if an error occurred.

List Files

- readdir()
  - syntax: *struct dirent *readdir(DIR* dirp);*

```
struct dirent {
    ino_t          d_ino;       /* Inode number */
    off_t          d_off;       /* Not an offset */
    unsigned short d_reclen;    /* Length of this record*/
    unsigned char  d_type;      /* Type of file */
    char           d_name[256]; /* Null-terminated filename*/
}
```

  - *d_name* is the file name.

Tips

- Cross platform machines
  - https://corner.cse.cuhk.edu.hk/fac/unix.html
  - Since files are synchronized automatically in our department machines, you need to put “server” and “client” under different folders when testing on cross platforms.
  - “ifconfig”: check IP address
- Make sure you can transfer large files.
  - We assume the file size is at most 1 GiB.

Thank you!