

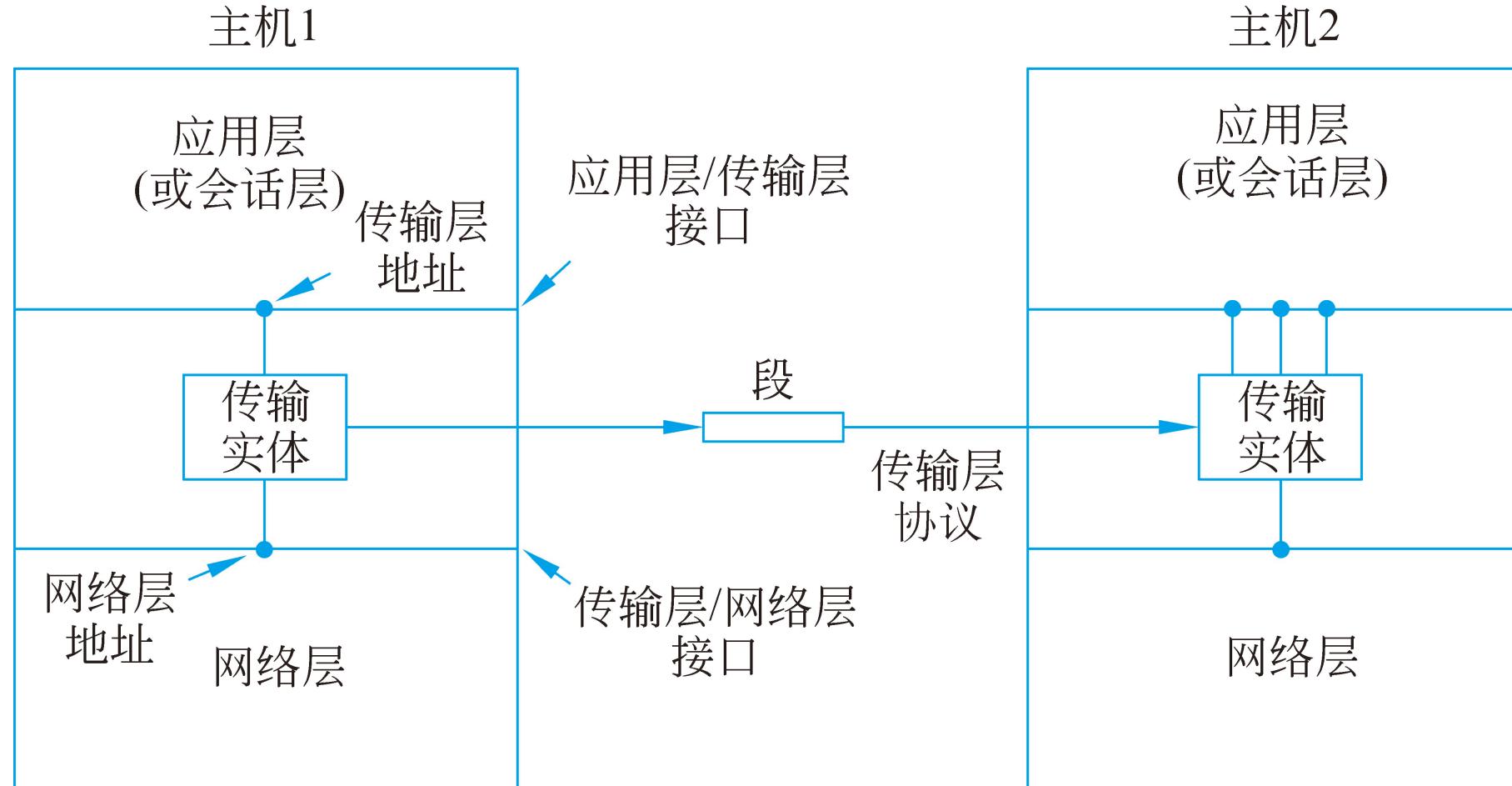
第三章 网络编程

授课教师：李彤

中国人民大学



传输层的位置

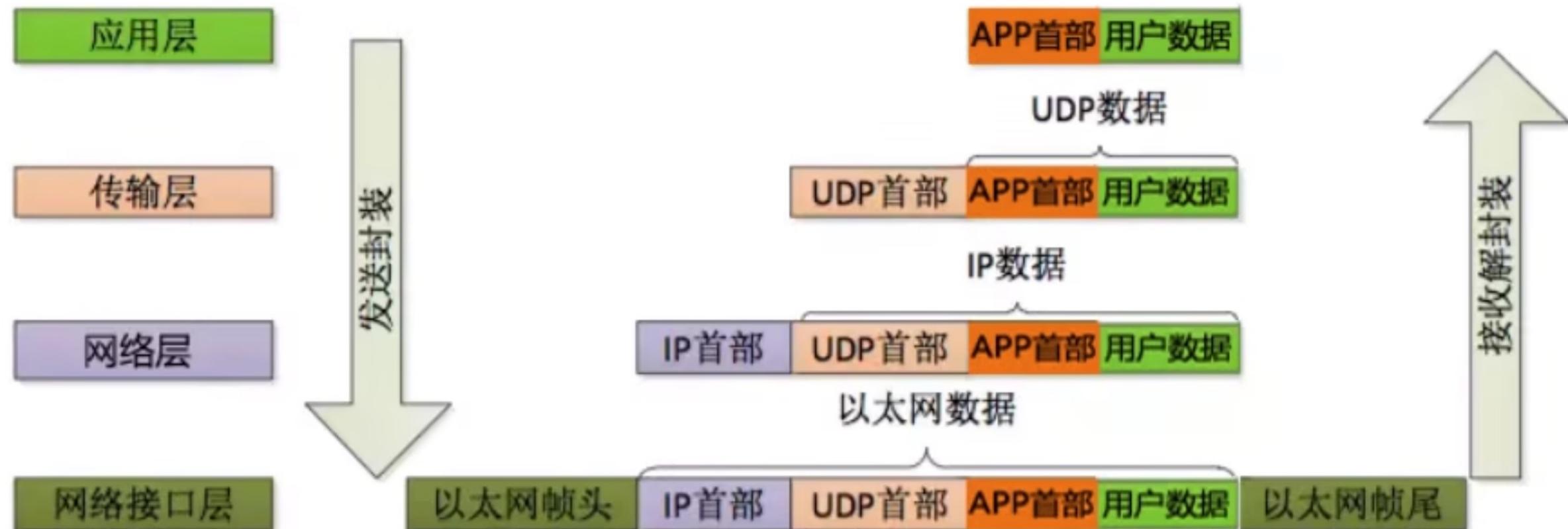




传输层的位置



中國人民大學
RENMIN UNIVERSITY OF CHINA





- 最低限度的传输服务：
 - 将终端-终端的数据交付扩展到进程-进程的数据交付
 - 报文检错
- 增强服务：
 - 可靠数据传输
 - 流量控制
 - 拥塞控制
- 因特网传输层通过UDP协议和TCP协议，向应用层提供两种不同的传输服务：
 - UDP协议：仅提供最低限度的传输服务
 - TCP协议：提供基础服务和增强服务



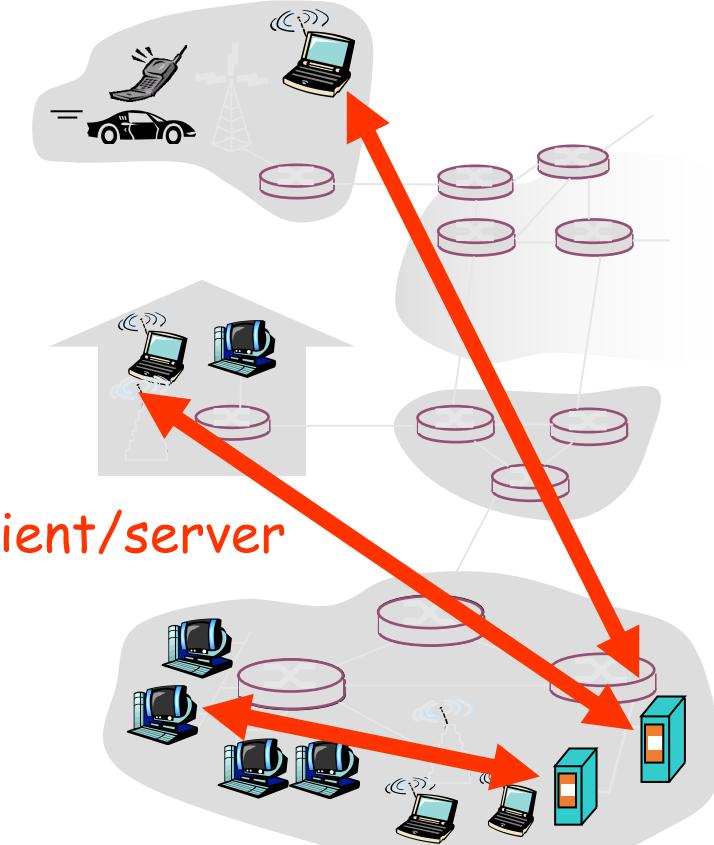
如何创建一个网络应用？

➤ 如何创建一个网络应用：

- 编写一个分布式程序，使其运行在不同的端系统上，并通过网络通信

➤ 选择一种应用程序体系结构：

- 传统及主流的是客户-服务器体系结构（C/S）
- 有一台总是在线的主机，运行一个**服务器程序**（server），服务器主机具有永久的、众所周知的地址
- 用户终端上运行一个**客户程序**（client），需要时主动与服务器程序通信，请求服务
- 客户只与服务器通信，客户之间不通信





进程如何标识自己



中國人民大學
RENMIN UNIVERSITY OF CHINA

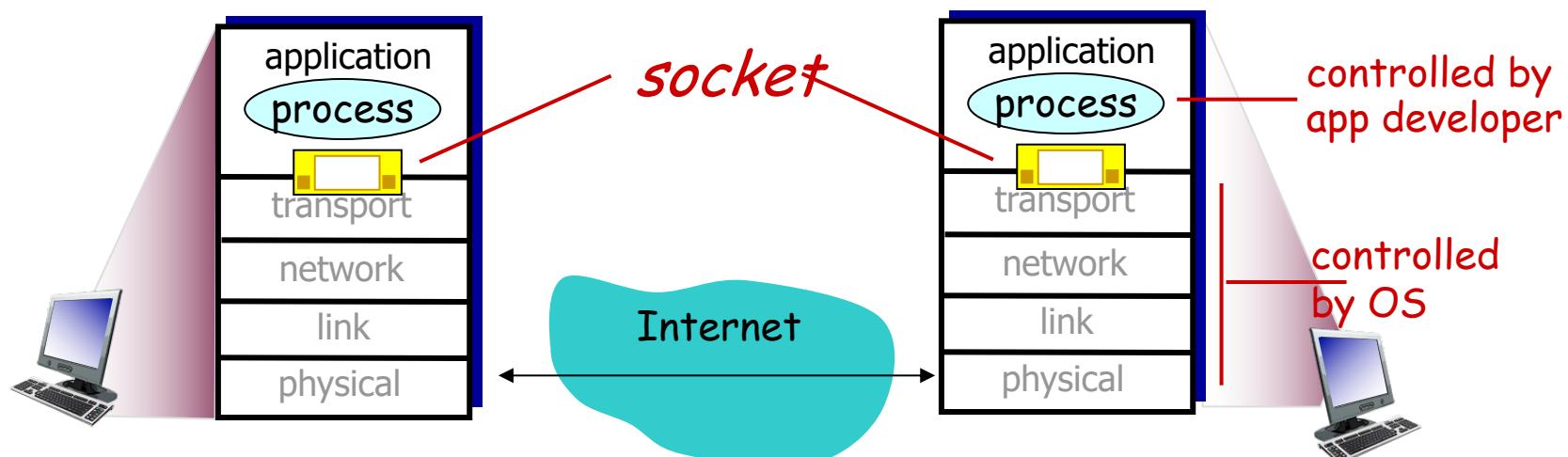
- 每个进程需要一个**标识**，以便其它进程能够找到它
 - **问题**：可以用进程所在主机的地址来标识进程吗？
 - **回答**：不能！因为同一个主机上可能运行着许多进程
- 端口号（port number）：
 - 端口号被用来区分同一个主机上的不同进程
- 因此，**进程标识包括**：
 - 主机地址
 - 主机上与该进程关联的端口号



不同终端上的进程如何通信？

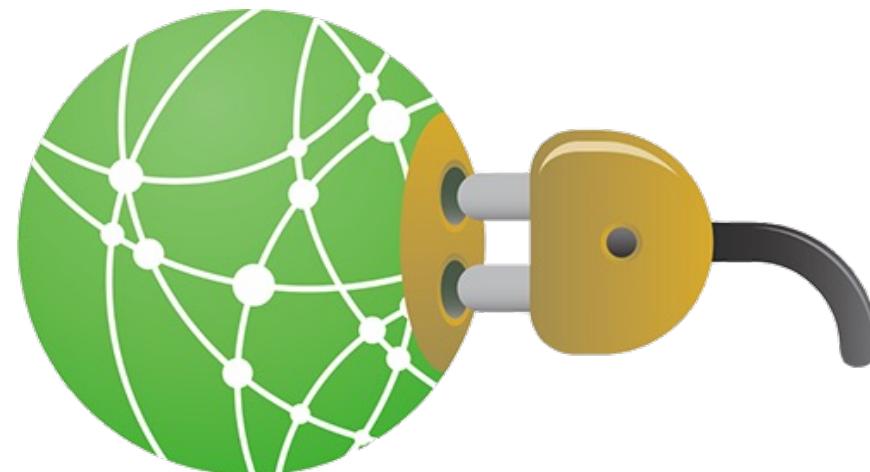


- 设想在应用程序和网络之间存在一扇“门”：
 - 需要发送报文时：发送进程将报文推到门外
 - 门外的运输设施（因特网）将报文送到接收进程的门口
 - 需要接收报文时：接收进程打开门，即可收到报文
- 在TCP/IP网络中，这扇“门”称为**套接字（socket）**，是应用层和传输层的接口，也是应用程序和网络之间的API





- Socket 套接字由ARPA资助UC Berkeley的一个研究组研发
- 目的是将 TCP/IP 协议相关软件移植到 UNIX 类系统中,设计者开发了一个接口，形成了 Socket 套接字
- Linux 系统采用了 Socket 套接字，成为事实标准



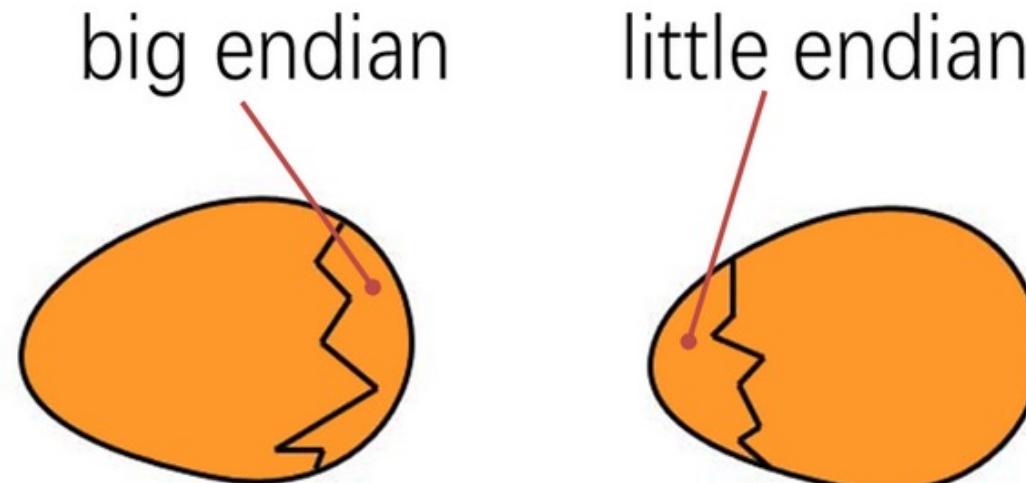


字节序



中國人民大學
RENMIN UNIVERSITY OF CHINA

- 计算机中通常采用的字节存储机制主要有两种：大端（Big endian）和小端（Little endian）





- Little-Endian -> 主机字节序 (小端)
 - 数据的**低位字节**存储到内存的**低地址位**，数据的**高位字节**存储到内存的**高地址位**
 - 我们使用的 PC 机，数据的存储默认使用的是小端
- Big-Endian -> 网络字节序 (大端)
 - 数据的**低位字节**存储到内存的**高地址位**，数据的**高位字节**存储到内存的**低地址位**
 - 套接字通信过程中操作的数据都是大端存储的，包括：接收/发送的数据、IP地址、端口



字节序



- 有两个分别需要4个字节存储的整数，使用16进制表示这两个数，即0x12345678和0x11223344。采用以下方式存储：

0x78	0x56	0x34	0x12	0x44	0x33	0x22	0x11
------	------	------	------	------	------	------	------

“小端 (Little endian) ”

内存地址增长方向



0x12	0x34	0x56	0x78	0x11	0x22	0x33	0x44
------	------	------	------	------	------	------	------

大端 (Big endian) ”

内存地址增长方向



对于单字符来说是没有字节序问题的



- BSD Socket 提供了转换接口，方便程序员使用
 - 从主机字节序到网络字节序的转换函数： htons、 htonl
 - 从网络字节序到主机字节序的转换函数： ntohs、 ntohl

```
#include <arpa/inet.h>
// u:unsigned
// 16: 16位, 32:32位
// h: host, 主机字节序
// n: net, 网络字节序
// s: short
// l: int
```

```
// 这套api主要用于 网络通信过程中 IP 和 端口 的 转换
// 将一个短整形从主机字节序 -> 网络字节序
uint16_t htons(uint16_t hostshort);
// 将一个整形从主机字节序 -> 网络字节序
uint32_t htonl(uint32_t hostlong);

// 将一个短整形从网络字节序 -> 主机字节序
uint16_t ntohs(uint16_t netshort)
// 将一个整形从网络字节序 -> 主机字节序
uint32_t ntohl(uint32_t netlong);
```



- IP 地址一般通过字符串描述，因此，通常使用专用转换API
 - 将一个字符串类型的 IP 地址进行大小端转换

```
// 主机字节序的IP地址转换为网络字节序
// 主机字节序的IP地址是字符串， 网络字节序IP地址是整形
int inet_pton(int af, const char *src, void *dst);
```

```
#include <arpa/inet.h>
// 将大端的整形数， 转换为小端的点分十进制的IP地址
const char *inet_ntop(int af, const void *src, char *dst, socklen_t size);
```

- 参数:
 - af: 地址族 (IP 地址的家族包括 ipv4 和 ipv6) 协议
 - AF_INET: ipv4 格式的 ip 地址
 - AF_INET6: ipv6 格式的 ip 地址
 - src: 传入参数，对应要转换的点分十进制的 ip 地址: 192.168.1.100
 - dst: 传出参数，函数调用完成，转换得到的大端整形 IP 被写入到这块内存
 - 返回值：成功返回 1，失败返回 0 或者 -1
- 参数:
 - af: 地址族协议
 - AF_INET: ipv4 格式的 ip 地址
 - AF_INET6: ipv6 格式的 ip 地址
 - src: 传入参数，这个指针指向的内存中存储了大端的整形 IP 地址
 - dst: 传出参数，存储转换得到的小端的点分十进制的 IP 地址
 - size: 修饰 dst 参数的，标记 dst 指向的内存中最多可以存储多少个字节



套接字地址：通用地址sockaddr



中國人民大學
RENMIN UNIVERSITY OF CHINA

- sockaddr 是一个套接字API接口的通用地址类型
- sa_family 表示地址族协议，一般采用 “AF_XXX” 形式
 - AF_INET: ipv4 格式的 ip 地址
 - AF_INET6: ipv6 格式的 ip 地址
 - AF_UNIX: unix 格式的ip地址
- sa_data 表示地址：端口(2字节) + IP地址(4字节) + 填充(8字节)

```
// 在写数据的时候不好用
struct sockaddr {
    sa_family_t sa_family;          // 地址族协议, ipv4
    char        sa_data[14];         // 端口(2字节) + IP地址(4字节) + 填充(8字节)
}
```



套接字地址：IPv4地址sockaddr_in



中國人民大學
RENMIN UNIVERSITY OF CHINA

- sin_family : AF_INET(IPv4) , 编程时实际使用地址类型
- sin_port : 存放端口号(按照网络字节序存储)
- sin_addr : 存放32位IP地址(无符号整数)
- sin_zero : 为与sockaddr大小兼容而保留的空字节

相比sockaddr
更方便使用

```
struct sockaddr_in { //struct to hold an address
    sa_family_t sin_family; //always AF_INET
    in_port_t sin_port; //protocol port number: uint16_t
    struct in_addr sin_addr; //IP address
    char sin_zero[8]; //unused(set to zero)
};

struct in_addr {
    in_addr_t s_addr; //IPv4 address (uint32_t)
};
```



套接字接口：服务原语



中國人民大學
RENMIN UNIVERSITY OF CHINA

原语	发出的数据包	含义
LISTEN	(无)	阻塞，直到某个进程试图与之连接
CONNECT	CONNECTION REQ	主动尝试建立一个连接
SEND	DATA	发送信息
RECEIVE	(无)	阻塞，直到有一个DATA 数据包到达
DISCONNECT	DISCONNECTION REQ	请求释放连接



Berkeley套接字



中國人民大學
RENMIN UNIVERSITY OF CHINA

原语	含义
SOCKET	创建一个新通信端点
BIND	将套接字与一个本地地址关联
LISTEN	声明愿意接受连接；给出队列大小
ACCEPT	被动创建一个进来的连接
CONNECT	主动创建一个连接
SEND	通过连接发送一些数据
RECEIVE	从连接上接收一些数据
CLOSE	释放连接



创建套接字 : socket()



- 客户或服务器调用socket()创建本地套接字，返回套接字描述符
- domain指明网络层地址类型
- type指明传输层协议：
 - SOCK_STREAM代表TCP字节流
 - SOCK_DGRAM代表UDP数据报

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

```
int socket(
    int domain, /* AF_UNIX, AF_INET, AF_INET6. */
    int type,   /* SOCK_STREAM, SOCK_DGRAM */
    int protocol); /* usually zero */
/* Returns file descriptor or -1 on error (sets errno) */
```

函数的返回值是一个文件描述符，通过它以操作内核的某块内存，网络通信基于该文件描述符完成

使用套接字通信函数需要包含头文件 `<arpa/inet.h>`，就不用再包含`<sys/socket.h>` 了



绑定套接字地址 : bind()



中國人民大學
RENMIN UNIVERSITY OF CHINA

- bind() 用来文件描述符 sockfd 与本地地址 addr 绑定
- 客户程序不需要调用bind()，操作系统将为其在1024 ~ 5000之间分配一个端口号

```
// 将文件描述符和本地的IP与端口进行绑定
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

- 参数:
 - sockfd: 监听的文件描述符, 通过 socket () 调用得到的返回值
 - addr: 传入参数, 要绑定的 IP 和端口信息需要初始化到这个结构体中, **IP和端口要转换为网络字节序**
 - addrlen: 参数 addr 指向的内存大小, sizeof (struct sockaddr)
- 返回值: 成功返回 0, 失败返回 -1



监听套接字地址：listen()



- listen() 用来给监听的套接字设置监听
- backlog设定了监听队列长度
- 客户程序不需要调用listen()

```
// 给监听的套接字设置监听
int listen(int sockfd, int backlog);
```

- 参数：
 - sockfd: 文件描述符，可以通过调用 socket () 得到，在监听之前必须要绑定 bind ()
 - backlog: 同时能处理的最大连接要求，最大值为 128
 - 返回值：函数调用成功返回 0，调用失败返回 -1



接受套接字地址：accept()



中國人民大學
RENMIN UNIVERSITY OF CHINA

- accept() 用来等待并接受客户端的连接请求, 建立新的连接, 会得到一个新的文件描述符(通信的文件描述符)
- 客户程序不需要调用accept()

```
// 等待并接受客户端的连接请求，建立新的连接，会得到一个新的文件描述符(通信的)
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

- 参数：
 - sockfd: 监听的文件描述符
 - addr: 传出参数, 里边存储了建立连接的客户端的地址信息
 - addrlen: 传入传出参数, 用于存储 addr 指向的内存大小
- 返回值：函数调用成功，得到一个文件描述符，用于和建立连接的这个客户端通信。调用失败返回 -1

Accept()是一个
阻塞函数



接收数据套接字地址：read/recv()



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
ssize_t read(int sockfd, void *buf, size_t size);  
ssize_t recv(int sockfd, void *buf, size_t size, int flags);
```

- 参数:

- sockfd: 用于通信的文件描述符, accept () 函数的返回值
- buf: 指向一块有效内存, 用于存储接收是数据
- size: 参数 buf 指向的内存的容量
- flags: 特殊的属性, 一般不使用, 指定为 0

- 返回值:

- 大于 0: 实际接收的字节数
- 等于 0: 对方断开了连接
- -1: 接收数据失败了

read/recv()也是
阻塞函数



发送数据套接字地址：write/send()



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
ssize_t write(int fd, const void *buf, size_t len);  
ssize_t send(int fd, const void *buf, size_t len, int flags);
```

- › 参数：
 - fd: 通信的文件描述符, accept () 函数的返回值
 - buf: 传入参数, 要发送的字符串
 - len: 要发送的字符串的长度
 - flags: 特殊的属性, 一般不使用, 指定为 0
- › 返回值：
 - 大于 0: 实际发送的字节数, 和参数 len 是相等的
 - -1: 发送数据失败了

write/send()也
是阻塞函数



连接套接字地址：connect()

- connect() 用来与服务端建立连接
- 仅TCP客户程序调用connect()

```
// 成功连接服务器之后，客户端会自动随机绑定一个端口  
// 服务器端调用accept()的函数，第二个参数存储的就是客户端的IP和端口信息  
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

- 参数：
 - sockfd: 通信的文件描述符，通过调用 socket () 函数就得到了
 - addr: 存储了要连接的服务器端的地址信息: IP 和 端口，这个 IP 和端口也需要转换为大端然后再赋值
 - addrlen: addr 指针指向的内存的大小 sizeof (struct sockaddr)
- 返回值：连接成功返回 0，连接失败返回 -1



三次握手的过程
导致connect()
是阻塞函数



关闭套接字 : close()



中國人民大學
RENMIN UNIVERSITY OF CHINA

- 客户和服务器调用close()关闭一个套接字
- 当实参fd为TCP套接字描述符时：
 - 调用close()会引起本地进程向远程进程发送关闭连接的消息
- 当实参fd为UDP套接字描述符时：
 - 调用close()会引起为此描述符分配的资源被内核释放

```
#include <unistd.h>
```

```
int close(int fd);
/* Returns 0 if OK or -1 on error */
```



编程实例：回音服务



中國人民大學
RENMIN UNIVERSITY OF CHINA

1. 客户程序将一行字符（数据），发送给服务器
2. 服务器接收数据，然后将收到的数据回送给客户
3. 客户接收回送的数据，在屏幕上显示出来



使用UDP套接字实现回音服务



中國人民大學
RENMIN UNIVERSITY OF CHINA

- 客户或服务器调用**sendto()**发送数据：
 - 数据在buff中，长度为nbytes字节
 - 对方通信进程的地址在to中
- 客户或服务器调用**recvfrom()**接收数据：
 - 收到的数据放在buff中
 - 对方通信进程的地址在from中

```
#include <sys/socket.h>
ssize_t sendto(
    int socket_fd,
    const void *buff,
    size_t nbytes,
    int flags, /* usually 0 */
    const struct sockaddr *to,
    socklen_t *addrlen,
);
/*Return number of bytes written
if OK or -1 on error*/
```

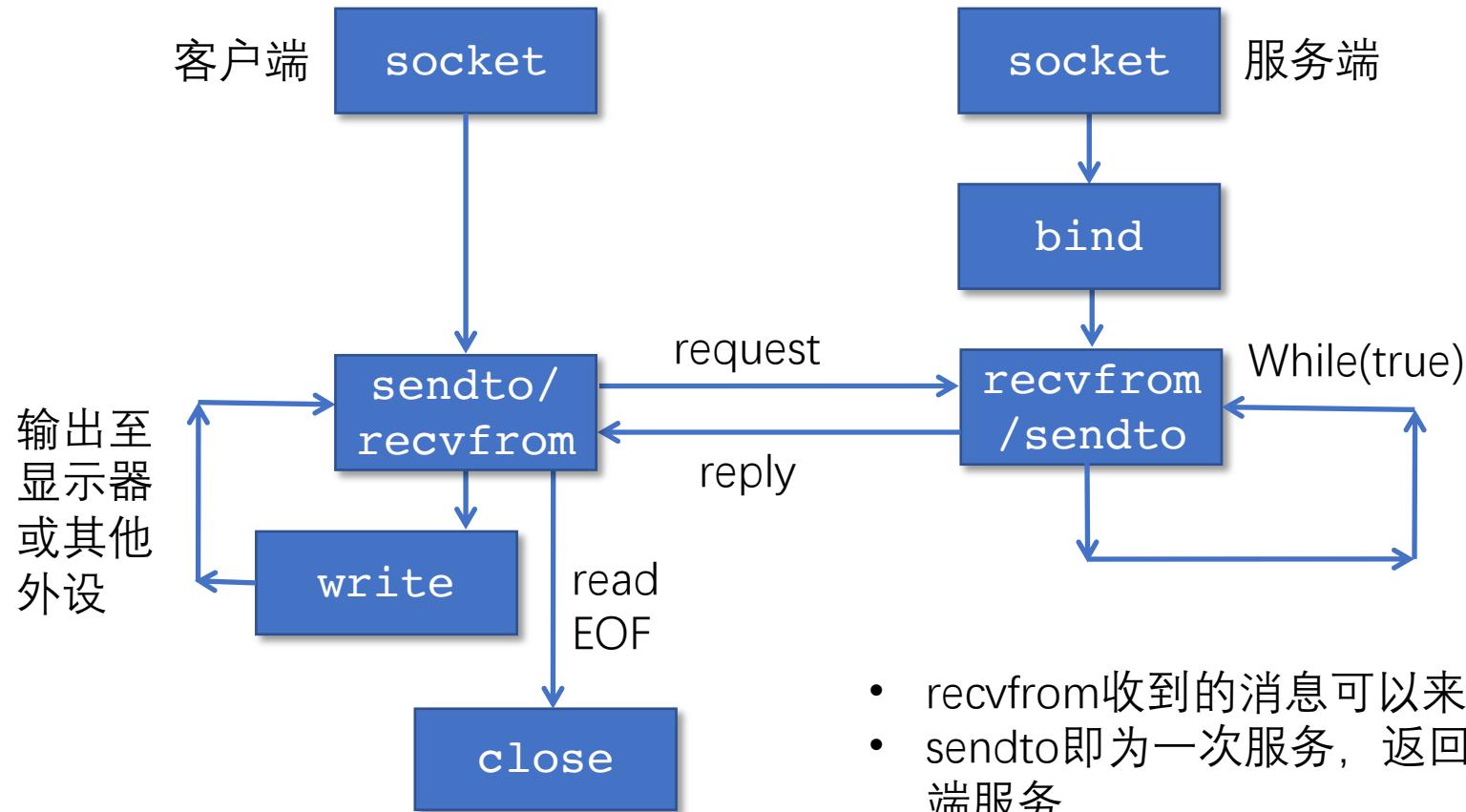
```
#include <sys/socket.h>
ssize_t recvfrom(
    int socket_fd,
    void *buff,
    size_t nbytes,
    int flags, /* usually 0 */
    struct sockaddr *from,
    socklen_t *addrlen,
);
/*Return number of bytes read if
OK or -1 on error*/
```



基于UDP的套接字通信流程



中國人民大學
RENMIN UNIVERSITY OF CHINA



- **recvfrom**收到的消息可以来自不同的客户端；
- **sendto**即为一次服务，返回后继续为下个客户端服务



UDP回音服务端代码：main



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
1 int main(int argc, char **argv){  
2     char mesg[MAXLINE];  
3     int sockfd, n, len;  
4     struct sockaddr_in cliaddr, servaddr;  
5  
6     sockfd = socket(AF_INET, SOCK_DGRAM, 0);  
7     bzero(&servaddr, sizeof(servaddr));  
8     servaddr.sin_family = AF_INET;  
9     servaddr.sin_addr.s_addr = htonl(INADDR_ANY);  
10    servaddr.sin_port = htons(SERV_PORT);  
11  
12    bind(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr));  
13  
14    for ( ; ; ) {  
15        len = sizeof(cliaddr);  
16        n = recvfrom(sockfd, mesg, MAXLINE, 0, cliaddr, &len);  
17        sendto(sockfd, mesg, n, 0, cliaddr, len);  
18    }  
19 }
```



UDP回音客户端代码：Part 1



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
1 int main(int argc, char **argv){  
2     int sockfd, n;  
3     struct sockaddr_in servaddr;  
4     char sendline[MAXLINE], recvline[MAXLINE + 1];  
5     if (argc != 2){  
6         fprintf(stderr, %s\n, "usage: echoCli <IPaddress>");  
7         exit(EXIT_FAILURE);  
8     }  
9     sockfd = socket(AF_INET, SOCK_DGRAM, 0);  
10    bzero(&servaddr, sizeof(servaddr));  
11    servaddr.sin_family = AF_INET;  
12    inet_pton(AF_INET, argv[1], &servaddr.sin_addr);  
13    servaddr.sin_port = htons(SERV_PORT);
```



UDP回音客戶端代码 : Part 2



中國人民大學
RENMIN UNIVERSITY OF CHINA

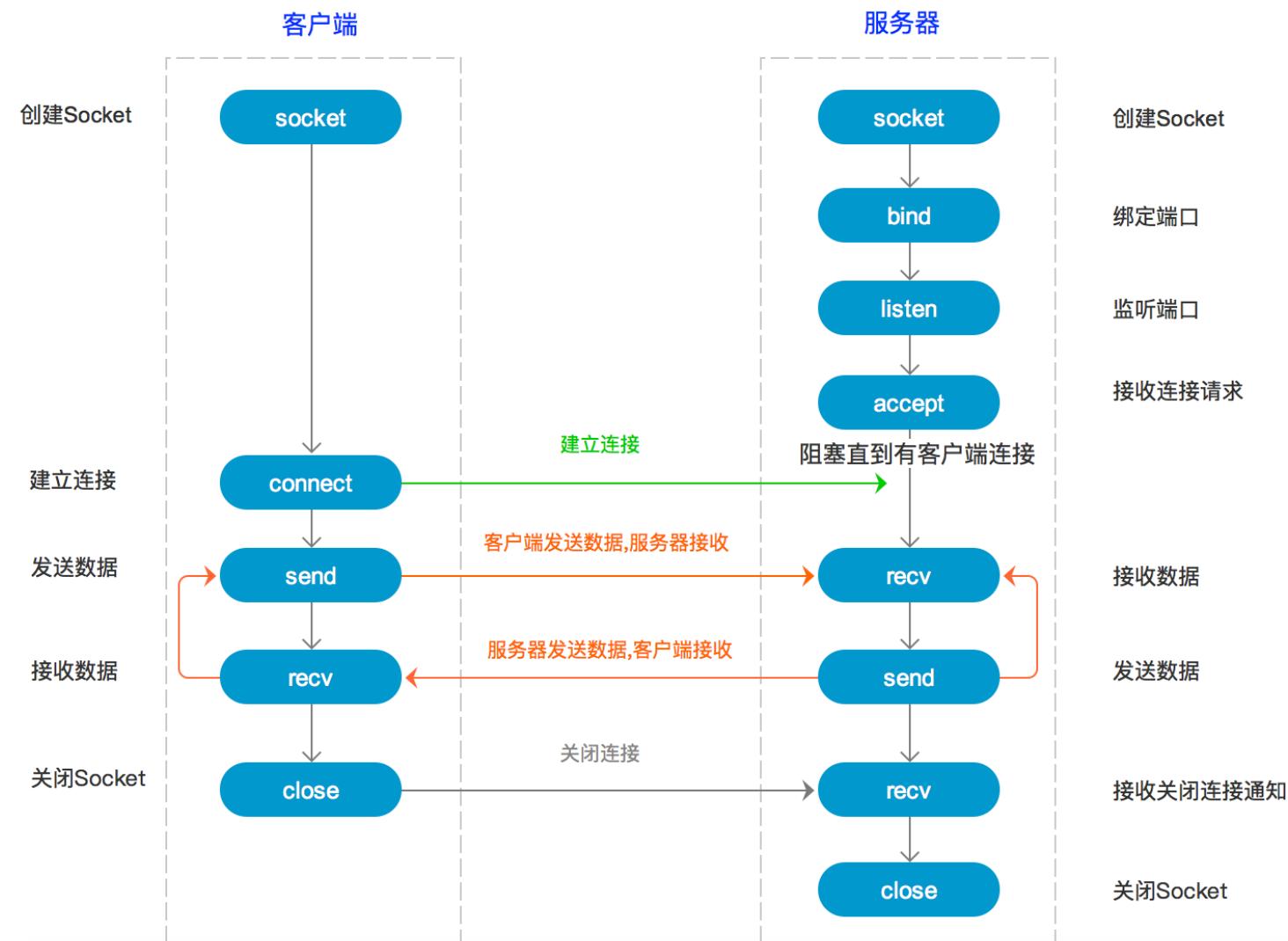
```
14 while (fgets(sendline, MAXLINE, stdin) != NULL) {  
15     sendto(sockfd, sendline, strlen(sendline), 0,  
              (struct sockaddr *)servaddr, sizeof(servaddr));  
16     n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);  
17     recvline[n] = 0;  
18     fputs(recvline, stdout);  
19 }  
20 exit(0);  
21 }
```



使用TCP套接字的通信流程



中國人民大學
RENMIN UNIVERSITY OF CHINA

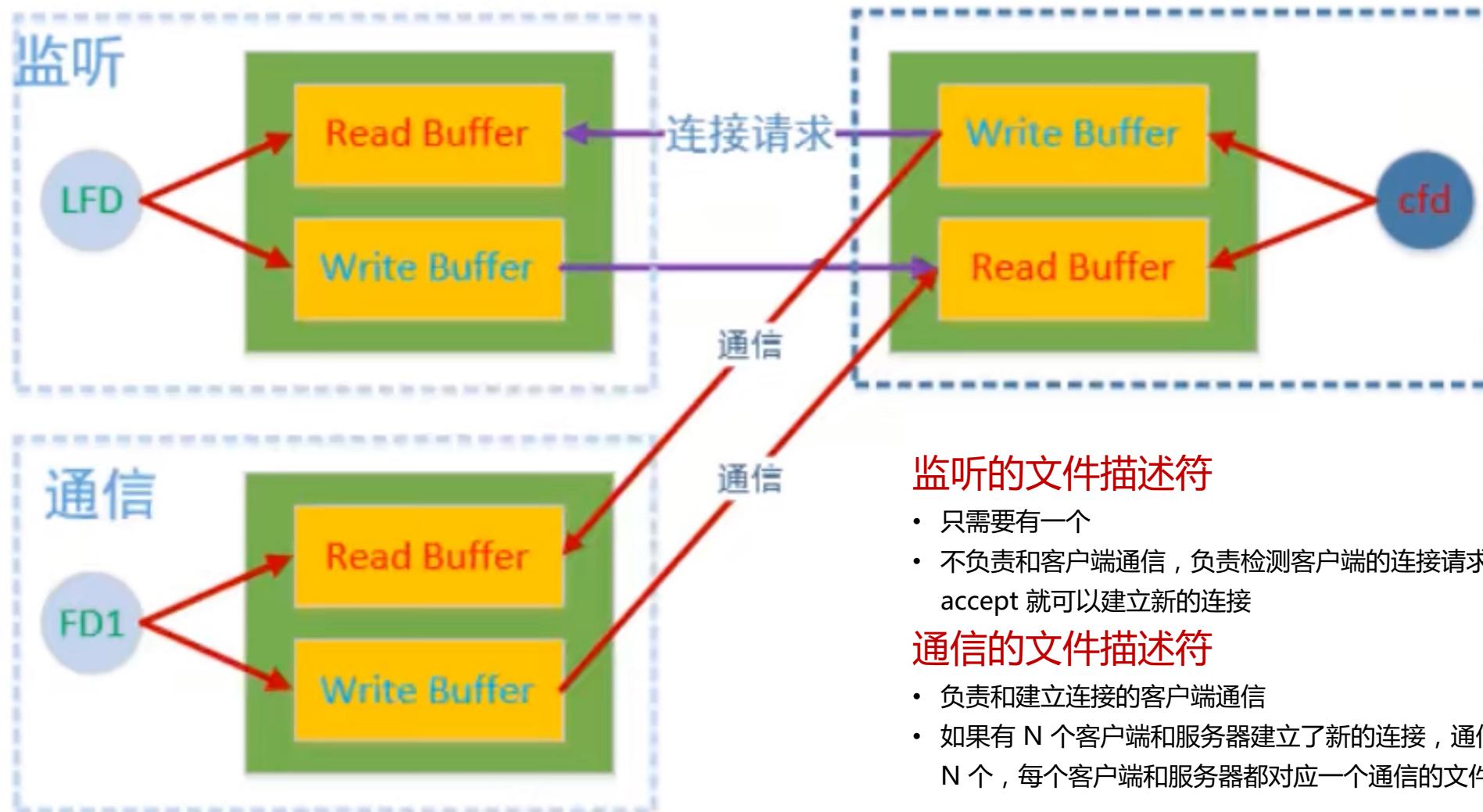




服务器使用多个套接字服务客户



中國人民大學
RENMIN UNIVERSITY OF CHINA



监听的文件描述符

- 只需要有一个
- 不负责和客户端通信，负责检测客户端的连接请求，检测到之后调用 accept 就可以建立新的连接

通信的文件描述符

- 负责和建立连接的客户端通信
- 如果有 N 个客户端和服务建立了新的连接，通信的文件描述符就有 N 个，每个客户端和服务都对应一个通信的文件描述符



TCP回音服务端代码 (1)



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
// server.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>

int main()
{
    // 1. 创建监听的套接字
    int lfd = socket(AF_INET, SOCK_STREAM, 0);
    if(lfd == -1)
    {
        perror("socket");
        exit(0);
    }
}
```



TCP回音服务端代码 (2)



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
// 2. 将socket()返回值和本地的IP端口绑定到一起
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons(10000); // 大端端口
// INADDR_ANY代表本机的所有IP，假设有三个网卡就有三个IP地址
// 这个宏可以代表任意一个IP地址
// 这个宏一般用于本地的绑定操作
addr.sin_addr.s_addr = INADDR_ANY; // 这个宏的值为0 == 0.0.0.0
int ret = bind(lfd, (struct sockaddr*)&addr, sizeof(addr));
if(ret == -1)
{
    perror("bind");
    exit(0);
}
```



```
// 3. 设置监听
ret = listen(lfd, 128);
if( ret == -1)
{
    perror("listen");
    exit(0);
}
```



TCP回音服务端代码 (4)



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
// 4. 阻塞等待并接受客户端连接
struct sockaddr_in cliaddr;
int clilen = sizeof(cliaddr);
int cfd = accept(lfd, (struct sockaddr*)&cliaddr, &clilen);
if(cfd == -1)
{
    perror("accept");
    exit(0);
}

// 打印客户端的地址信息
char ip[24] = {0};
printf("客户端的IP地址: %s, 端口: %d\n",
inet_ntop(AF_INET, &cliaddr.sin_addr.s_addr, ip, sizeof(ip)),
 ntohs(cliaddr.sin_port));
```



TCP回音服务端代码 (5)



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
// 5. 和客户端通信
while(1)
{
    // 接收数据
    char buf[1024];
    memset(buf, 0, sizeof(buf));
    int len = read(cfd, buf, sizeof(buf));
    if(len > 0)
    {
        printf("客户端say: %s\n", buf);
        write(cfd, buf, len);
    }
    else if(len == 0)
    {
        printf("客户端断开了连接...\n");
        break;
    }
    else
    {
        perror("read");
        break;
    }
}
```



TCP回音服务端代码 (6)



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
// 6. 关闭连接
close(cfd);
close(lfd);

return 0;
}
```



TCP回音客户端代码 (1)



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
// client.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>

int main()
{
    // 1. 创建通信的套接字
    int fd = socket(AF_INET, SOCK_STREAM, 0);
    if(fd == -1)
    {
        perror("socket");
        exit(0);
    }
```



```
// 2. 连接服务器
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons(10000); // 大端端口
inet_pton(AF_INET, "127.0.0.1", &addr.sin_addr.s_addr);

int ret = connect(fd, (struct sockaddr*)&addr,
sizeof(addr));
if(ret == -1)
{
perror("connect");
exit(0);
}
```



TCP回音客户端代码 (3)



中國人民大學
RENMIN UNIVERSITY OF CHINA

```
// 3. 和服务器端通信
int number = 0;
while(1)
{
    // 发送数据
    char buf[1024];
    sprintf(buf, "你好, 服务器...%d\n", number++);
    write(fd, buf, strlen(buf)+1);
    // 接收数据
    memset(buf, 0, sizeof(buf));
    int len = read(fd, buf, sizeof(buf));
    if(len > 0)
    {
        printf("服务器say: %s\n", buf);
    }
    else if(len == 0)
    {
        printf("服务器断开了连接...\n"); break;
    }
    else
    {
        perror("read"); break;
    }
    sleep(1); // 每隔1s发送一条数据
}
```



TCP回音客户端代码 (4)



中國人民大學
RENMIN UNIVERSITY OF CHINA

// 4. 关闭连接

```
close(fd);
```

```
return 0;
```

```
}
```



TCP回音演示



中國人民大學
RENMIN UNIVERSITY OF CHINA

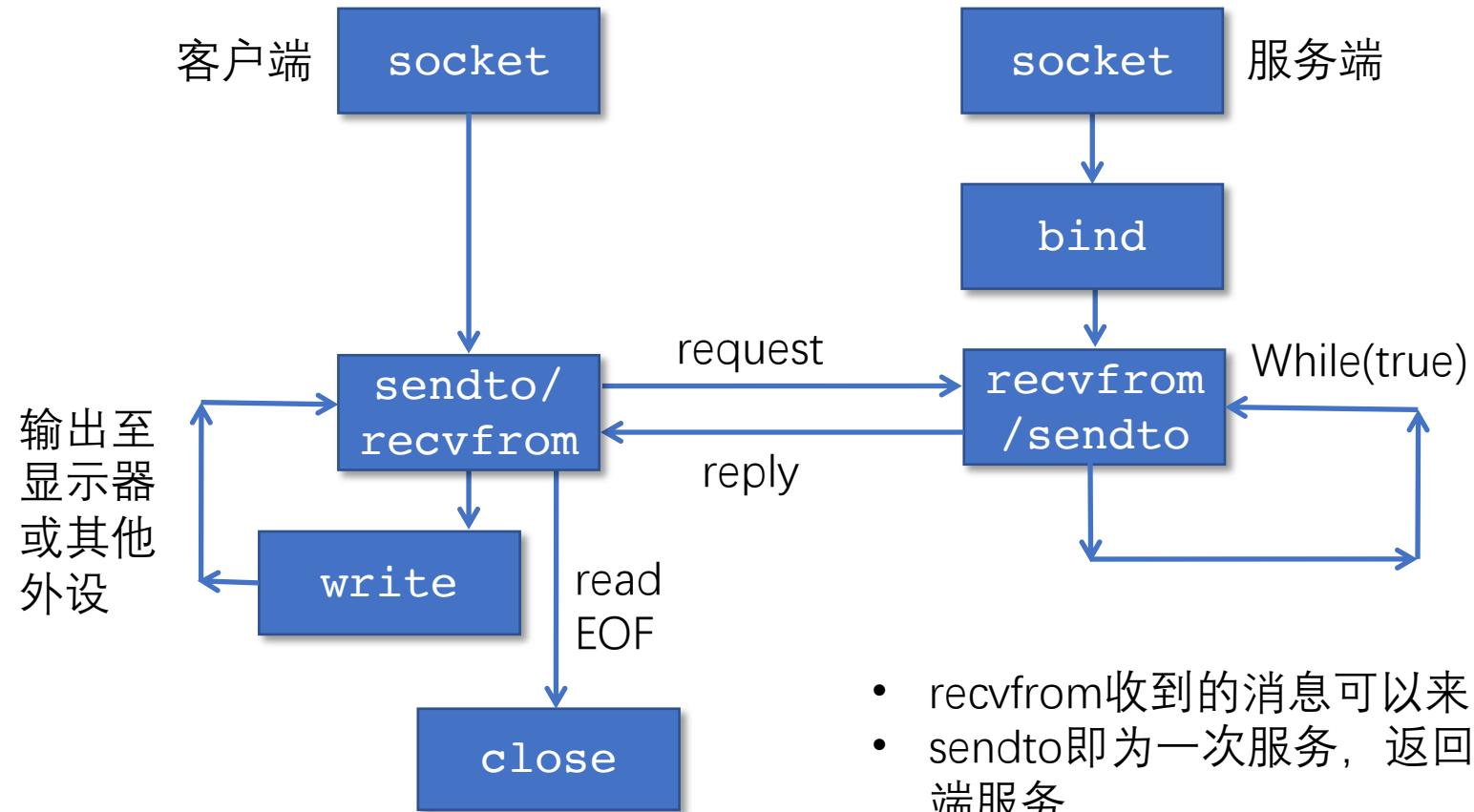
The image shows two terminal windows side-by-side. The left window is titled "network — client — 80x24" and the right window is titled "network — server — 80x24". Both windows have a title bar with a red close button, a yellow minimize button, and a green maximize button. The client window displays the following text:
[litong@bogon network % ./client
服务器say: 你好，服务器...0
服务器say: 你好，服务器...1
服务器say: 你好，服务器...2
服务器say: 你好，服务器...3
服务器say: 你好，服务器...4
服务器say: 你好，服务器...5
服务器say: 你好，服务器...6
服务器say: 你好，服务器...7
服务器say: 你好，服务器...8
服务器say: 你好，服务器...9
服务器say: 你好，服务器...10
The right window displays the following text:
客户端的IP地址: 127.0.0.1, 端口: 56313
客户端say: 你好，服务器...0
客户端say: 你好，服务器...1
客户端say: 你好，服务器...2
客户端say: 你好，服务器...3
客户端say: 你好，服务器...4
客户端say: 你好，服务器...5
客户端say: 你好，服务器...6
客户端say: 你好，服务器...7
客户端say: 你好，服务器...8
客户端say: 你好，服务器...9
客户端say: 你好，服务器...10



基于UDP的套接字通信流程



中國人民大學
RENMIN UNIVERSITY OF CHINA



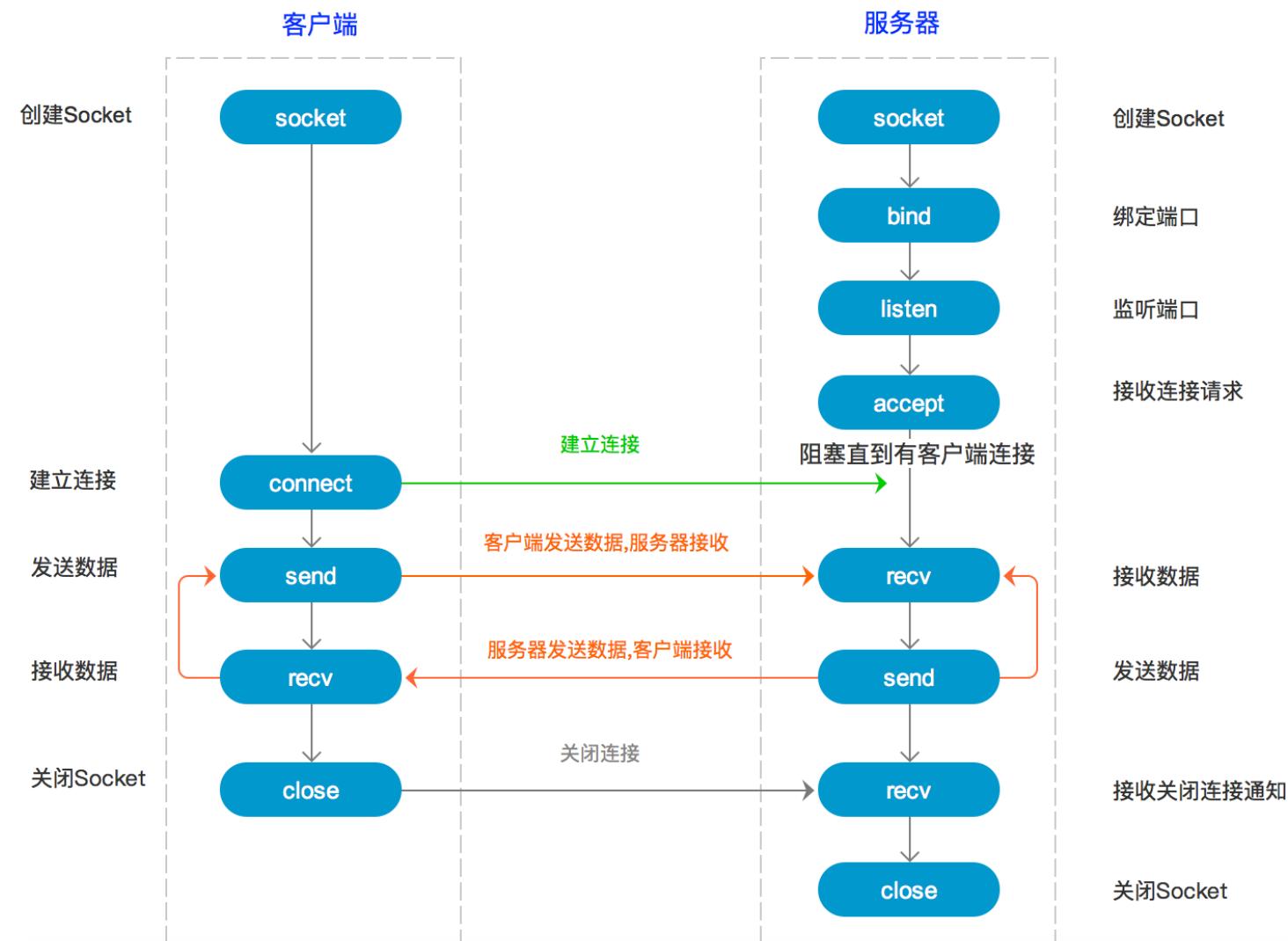
- `recvfrom`收到的消息可以来自不同的客户端；
- `sendto`即为一次服务，返回后继续为下个客户端服务



使用TCP套接字的通信流程



中國人民大學
RENMIN UNIVERSITY OF CHINA





➤ UDP套接字

- 使用<IP地址，端口号>二元组标识UDP套接字
- 服务器使用**一个套接字**服务所有客户

➤ TCP套接字

- 使用<源IP地址，目的IP地址，源端口号，目的端口号> 四元组标识连接套接字
- 服务器使用**一个监听套接字和多个连接套接字**服务多个客户，
每个连接套接字服务一个客户



思考：阻塞问题



中國人民大學
RENMIN UNIVERSITY OF CHINA

