从零编写基于 UDP 的通信程序

作者：zhaobingchun
原文链接：https://ld246.com/article/1602332229241
来源网站：链滴

为什么使用UDP

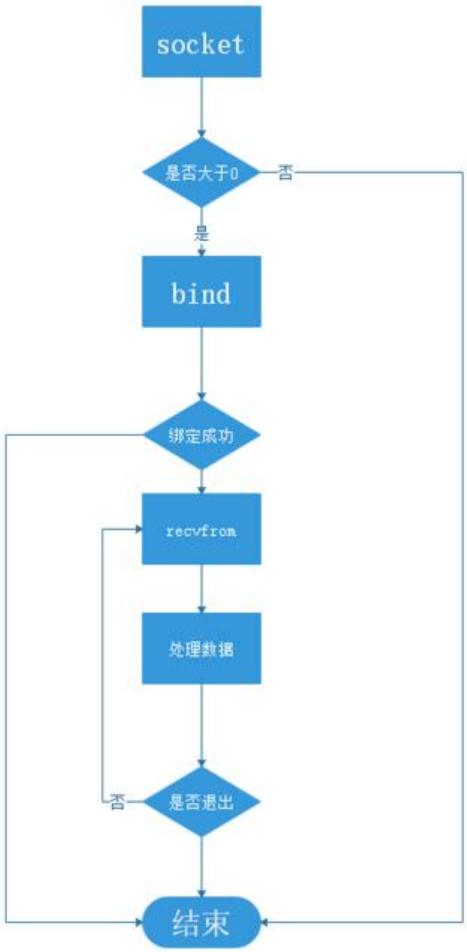
首先，我们知道TCP在弱网环境下有性能问题参考

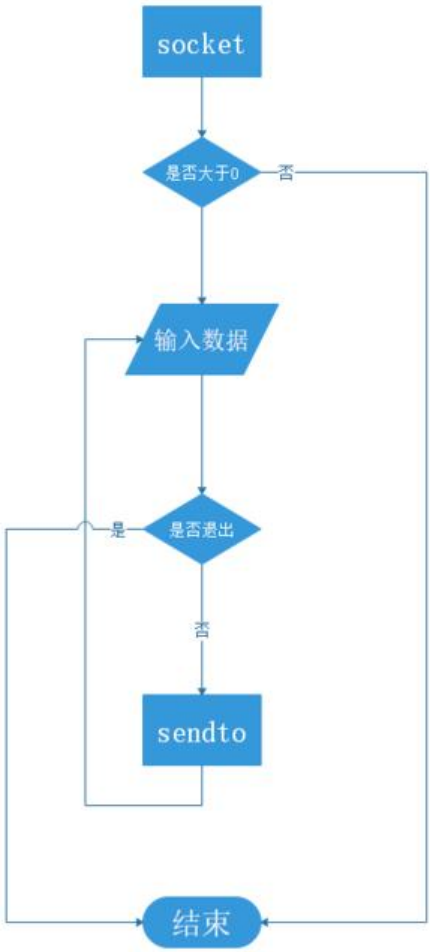
为什么 TCP 协议有性能问题

那么使用UDP有有哪些优势呢：

- UDP是无连接的，即通信时不需要创建连接（发送数据结束时也没有连接可以释放）所以减小了开和发送数据前的时延；
- UDP采用最大努力交付，不保证可靠交付，因此主机不需要维护复杂的连接状态；
- UDP是面向报文的，只在应用层交下来的报文前增加了首部后就向下交付IP层；
- UDP是无阻塞控制的，即使网络中存在阻塞，也不会影响发送端的发送频率；
- UDP支持一对一、一对多、多对一、多对多的交互通信；
- DUP的首部开销小，只有8个字节，它比TCP的20个字节的首部要短。

怎样实现UDP通信





udp通信流程图

首先实现server端:

```
int main(int argc, char **argv) {
    if(argc < 2) {
        printf("Usage: %s port\n", argv[0]);
        return -1;
    }
    int buf_size = BUF_SIZE;
    int len = sizeof(int);
    int s;
    struct sockaddr_in addr;

    unsigned short port = atoi(argv[1]);

    s = socket(AF_INET, SOCK_DGRAM, 0);  // 创建socket
    if(s < 0) {
        return -1;
    }

    // 服务端应该处理多个客户端请求，适当增大缓冲区，防止因为服务器处理慢导致丢包
    setsockopt(s, SOL_SOCKET, SO_SNDBUF, &buf_size, len);  // 设置发送buf
```

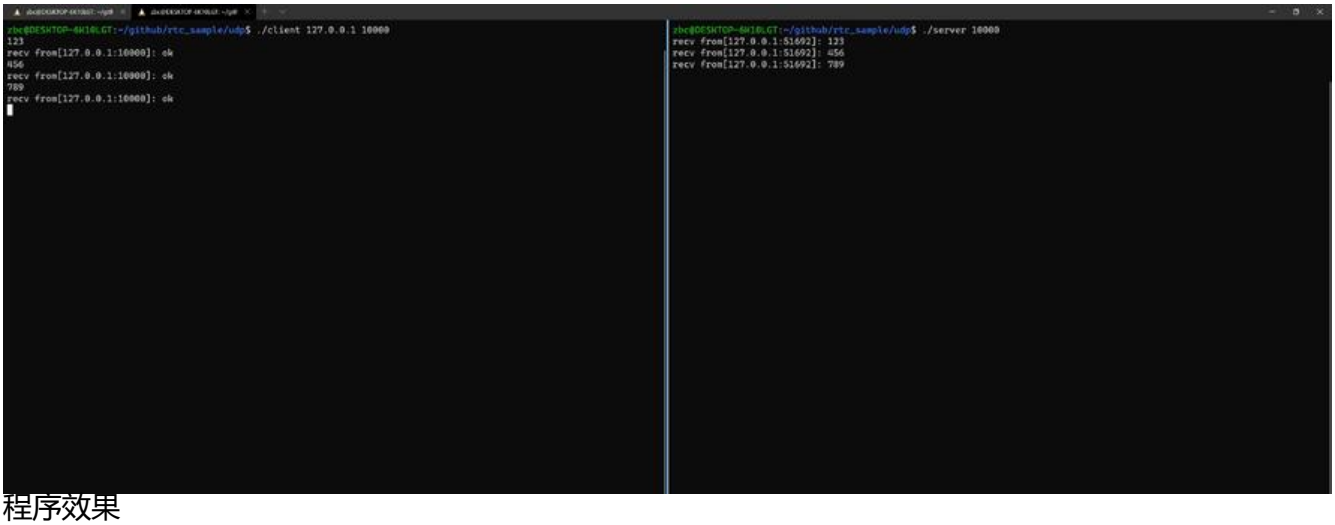
```
    setsockopt(s, SOL_SOCKET, SO_RCVBUF, &buf_size, len);  // 设置接收buf

    memset(&addr, 0, sizeof(struct sockaddr_in));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);

    // 绑定服务端端口号，客户端向此端口发送数据就可以
    if(bind(s, (struct sockaddr *)&addr, sizeof(struct sockaddr)) < 0 ){
        return -1;
    }
    // int flags;
    // if ((flags = fcntl(s, F_GETFL, 0)) < 0 || fcntl(s, F_SETFL, flags | O_NONBLOCK) < 0) {
    //     return -1;
    // }
    struct sockaddr_in recv_addr;
    socklen_t addr_len = sizeof(struct sockaddr_in);
    char buf[1024];
    while(1){
        memset(buf, 0, sizeof(buf));
        // 接收其他客户端消息
        int ret = recvfrom(s, buf, 1024, 0, (struct sockaddr*)&recv_addr, &addr_len);
        if(ret <= 0) {
            printf("server recv err: %d\n", ret);
            return -1;
        }
        char addr_s[INET_ADDRSTRLEN];
        inet_ntop(addr.sin_family, &recv_addr.sin_addr, addr_s, sizeof(addr_s));
        printf("recv from[%s:%d]: %s\n", addr_s, ntohs(recv_addr.sin_port), buf);
        // 回复客户端
        memset(buf, 0, sizeof(buf));
        sprintf(buf, "ok");
        sendto(s, buf, strlen(buf), 0, (struct sockaddr*)&recv_addr, addr_len);
    }

    return 0;
}
```

实现相应客户端

```
int main(int argc, char **argv) {
    if(argc < 3) {
        printf("Usage: %s addr port\n", argv[0]);
        return -1;
    }
    int s;
    struct sockaddr_in addr;

    unsigned short port = atoi(argv[2]);

    s = socket(AF_INET, SOCK_DGRAM, 0);  // 创建socket
    if(s < 0) {
        printf("create socket err %d\n", s);
        return -1;
```

```
    }
    memset(&addr, 0, sizeof(struct sockaddr_in));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = inet_addr(argv[1]);
    char buf[1024];

    struct sockaddr_in recv_addr;
    socklen_t addr_len = sizeof(struct sockaddr_in);
    while (1)
    {
        memset(buf, 0, 1024);
        scanf("%s", buf);
        if(strncmp(buf, "quit", 4) == 0) {
            return 0;
        }
        // 发送数据
        sendto(s, buf, strlen(buf), 0, (struct sockaddr*)&addr, sizeof(struct sockaddr));
        memset(buf, 0, sizeof(buf));
        // 响应服务端返回
        int ret = recvfrom(s, buf, 1024, 0, (struct sockaddr*)&recv_addr, &addr_len);
        if(ret <= 0) {
            printf("server recv err: %d\n", ret);
            return -1;
        }
        char addr_s[INET_ADDRSTRLEN];
        inet_ntop(addr.sin_family, &recv_addr.sin_addr, addr_s, sizeof(addr_s));
        printf("recv from[%s:%d]: %s\n", addr_s, ntohs(recv_addr.sin_port), buf);
    }
    return 0;
}
```

实现结果查看



程序效果

实现了客户端发送，服务器响应打印，并回复客户端，可以看到发送消息后，客户端正常收到服务器复的ok

总结

本例通过最简单的socket编程，实现了简单的udp通信，当然实际在服务器处理中还有很多复杂的逻，比如服务器目前是阻塞在udp的recvfrom，实际中服务器不可能只做这一件事，需要使用非阻塞方，或者使用多线程模式。另外，我们知道udp是不可靠传输，需要制定协议，做消息重排序，丢包重，或者使用fec技术恢复，这些留在下篇介绍。