网络程序设计实践

实验指导书

福州大学

数学与计算机科学学院

主讲教师：张浩

2018 年 9 月

本课程为《网络程序设计》课程的实践课，目的在于掌握虚拟实验环境的组建，通过实践加深理解网络编程的基本原理知识，特别是对 TCP 套接口、UDP 套接口、原始套接口、带外数据、阻塞与非阻塞 I/O 等网络编程知识的实践验证。在此基础上熟悉常见的应用层协议的实现。使得学生能够使用掌握的知识，独立设计和实现简单的网络通信工具。该课程的学习旨在加深学生对网络编程的了解和实践认识，提高学生 Linux 下代码学习和编辑能力，为其在程序设计领域的网络编程和开发等方面奠定基础。

1 实验一　搭建与熟悉网络应用程序设计平台

1.1 实验目的

（1）初步掌握虚拟机软件及 VMware 的概念和用处；
（2）初步掌握 Linux 中 SSH 服务的使用，SecureShell 客户端软件的使用；
（3）掌握 Linux 常用命令；
（4）掌握 Linux 下编程基础知识。

1.2 实验原理

1.2.1 虚拟机软件及 VMware 软件

虚拟机软件可在一台电脑上模拟出来若干台 PC，每台 PC 可以运行单独的操作系统而互不干扰，可以实现一台电脑“同时”运行几个操作系统，还可以将这几个操作系统连成一个网络。

例如如可以在一台电脑上安装了 Win2000 server，再在 Win2000 server 上安装虚拟机软件 VMWare，利用 VMWare 模拟出来 3 台 PC，在这 3 台 PC 上分别运行 RedHat、WinXP 和 Unix 操作系统。包括 Win2000 在内，这 4 个操作系统同时在一台电脑上运行，互不干扰，并且同在一个局域网内。

VMware 就是一款较为流行的虚拟机软件，它可以让我们在一台 PC 上实现一个虚拟的局域网络。主要用处与功能：

a) 为学习 Linux 和 Unix 操作系统的人提供了一个完全独立的学习平台；
b) 单机上实现网络程序（B/S,C/S）调试，这是双系统无法实现的；
c) 可以实时的进行系统间的切换；
d) VMware 软件可以运行在 Windows 和 Linux 平台上，支持的客户操作系统可以是各种 Linux，Unix 以及 Windows；
e) Linux 为客户机模拟各种硬件且与宿主机硬件无关，可以将虚拟机的整个系统拷贝到其他装有 VMware 软件的机器上运行，不需重新安装。

桥接模式设置网络：

vmware 网络设置：VM－>设置－>hardware->NIC 1：　Device status 选择：″connected″，″connect at power on″;Network connection 选择：“Bridged”。Edit->virtual Network Editor:Automatic Bridging（不选，也可以选择自动效果一样），Host Virtual Network Mapping :VMnet0 选择物理网卡，其他不用配置。

Linux 下：setup 命令中的网络配置，实现网络 IP 地址等信息配置，完成后 service network start.要求 eth0 启动为成功。要求宿主机和客户机位于一个网段。

vmware nat 网络设置：

NAT，自建一个小网络，不用公共 IP，不会和其他机器冲突 。

1.把你的虚拟网卡 VMnet8 设置为自动获得 IP、自动获得 DNS 服务器，启用。

2.把客户机的“本地连接”也设置为自动获得 IP、自动获得 DNS 服务器（在虚拟机中，右键“本地连接”－双击“Internet 协议”，看看是不是自动的吧！固定 IP 的也在这里改！）

3.将 vmware 的上网方式选为 NAT。

右键你要设置的虚拟机选“设置”（因为有的不止虚拟一台），在“硬件”中选“以

太网”，将右边的网络连接改为 NAT －确定。

4.点菜单栏里的“编辑”－选“虚拟网络设置”，先将“自动桥接”取消（去掉钩钩），再选“DHCP”开启 DHCP 服务，点“开始”－应用，再按同样的方法开启“NAT ”的功能。

5.最重要的是你的两个服务必须开启：VMware DHCP Service 和 VMware NATService. 具体操作如下：开始——设置——控制面板——管理工具——服务，确保 VMwareDHCP Service 和 VMware NAT Service 服务已经启动。

共有三个 IP 地址：宿主机的 IP，这个原来就设置好的（本地连接），大家都要通过他上网，不用管；虚拟网卡 VMnet8 是给宿主机提供的虚拟网卡，与客户机通信的，配置过程中要根据 NAT 提供的网段设置具体 IP ；客户机内部的 IP 地址，配置过程中要根据 NAT 提供的网段设置具体 IP ；后两个 IP 地址要位于同一个网段，网段地址信息在 NAT 设置时候可以看到包括掩码和网关地址。 也可以使用固定 ip 地址，不设置自动获取地址。自动获取 IP 地址过程 vmware 可能出现找不到网卡的情况，是 bug，通过修改脚本可以解决。

1.2.2 SSH 服务概述

使用 telnet 进行远程设备维护的时候，由于密码和通讯都是明文的，易受 sniffer 侦听，所以可采用 SSH 替代 telnet。SSH (Secure Shell)服务使用 tcp 22 端口，客户端软件发起连接请求后从服务器接受公钥，协商加密方法，成功后所有的通讯都是加密的。RedHat 系统中提供有该服务进程 sshd。

1.2.3 Linux 常用命令

帮助命令：man，在线显示手册。Linux 手册页主要有九个部分：用户指令（1）、系统调用（2）、程序库（3）、设备说明（4）、文件格式（5）、游戏（6）、杂项（7）、系统指令（8）、内核（9）。

文件与目录操作命令：cp,mv,rm,ln,mkdir,cd,pwd,ls,chmod 等；

系统管理命令：

mount,umount,useradd,userdel,passwd,ps,kill,tar,more,tail,head,service,netstat 等；

其他常见命令：echo,cal,date,clear,uname,who,whereis 等。

1.2.4 编程基础实践

（1）变元表

变元表是用来向执行的程序提供参数的，是一个指向字符串的指针数组。其长度可变，最大长度不应超过 5120 或 10?240 个字节。在 C 语言中变元表主要用 argc，argv 表示。

C 语言的 main 函数格式为：

```
main(int argc，char   argv[])
{
…
}
```

（2）环境表

在程序运行时，同接收变元表一样，也接收环境变量表。这个指针阵列由一个空指针结尾(元素个数未知)。三种方式

环境变量表可以作为程序的**第三个参数**来访问

```
main(int argc，char * argv[ ]，char *envp[ ])
{
…
```

}

外部变量 environ

C 库中的 **getenv 函数**

（3）进程 ID 和父进程 ID，进程的用户 ID 和有效用户 ID

getpid,getppid,getuid,geteuid。

（4）gcc 编译器

gcc(GNU cc)是一个编译器套件，gcc 支持三种语言的编译：C、C＋＋、Object C(C 语言的一种面向对象扩展)。gcc 可同时编译并连接 C(用 gcc)和 C++(用 g++)源程序。

编译命令：gcc hello.c －o hello

主要选项说明：

-c:只编译生成目标文件

-o FILE：生成指定的输出文件 FILE，用在生成可执行文件时.

-IDIRECTORY：指定额外的头文件搜索路径 DIRECTORY

在预处理和编译的时候使用，该路径将优先于系统的默认路径。在编译时提示找不到头文件，需要使用该选项告知编译器头文件的所在路径。

例如：hello.c 文件中有如下的两行：

#include <h1.h>

#include <h2.h>

其中 h1.h 在目录/usr/local/include 目录下，h2.h 在当前目录的上一层目录下.

编译命令：gcc -I/usr/local/include -I.. hello.c -o hello

-LDIRECTORY： 指定额外的函数库搜索路径 DIRECTORY

-lLIBRARY： 连接时搜索指定的函数库 LIBRARY。

例如：程序中使用了不在默认库中的调用，如在 hello.c 中调用 pthread_create()函数创建多线程，出错：undefined reference to ‘pthread_create’。因为没有告诉编译器在哪里能找到包含 pthread_create 的调用。pthread_create 在/usr/lib 目录下的库文件 libpthread.a(静态库)和 libpthread.so(动态库)中。编译命令：gcc -L/usr/lib -lpthread hello.c -o hello.c

-E：只运行 C 预编译器

例如： gcc -E hello.c -o hello.i(生成预处理后的 c 语言代码)

gcc -c hello.i（生成编译后目标代码 hello.o）

gcc hello.o -o hello(连接成可执行文件)

-static: 禁止使用共享连接，连接静态库，gcc 默认连接共享动态库。

连接静态库的文件要比连接动态库的文件大得多，但是它可以自给自足。如在没有安装 OpenGL 的系统上执行 OpenGL 程序。

-w ： 不生成任何警告信息

-Wall： 生成所有警告信息

GDB 程序调试工具简介

在调试状态下运行程序

设置断点，检查实时运行环境与变量

程序回溯，检查错误所在

单步执行

编译时参数设定： -g

gcc –g hello.c –o hello

启动执行 GDB

[root@localhost ABC-MRST]# gdb ./ hello

在 gdb 下运行程序

r 运行参数

(gdb) r ./a.txt

缺省命令为上一次执行的命令

如第一次执行 list

继续执行 list 时只需要直接回车（重复原命令）

gdb 常用命令

源码查看命令 list 或者 l

l —— 当前执行代码附近

l 代码行号 ——当前执行代码文件中相应行附近 10 行

l 文件名：代码行号 ——相应文件中相应行附近

(gdb) list main.cpp:10

l 文件名：函数名——相应文件中相应行附近

(gdb) list main.cpp:main

l + ——向前继续查看

l - ——向后继续查看

l 函数名——相应函数附近

list dataPreparetion(char*)

l first,last——从 first 到 last 行

(gdb) list 10,45

设置断点命令 break 或者 b

b 代码行号

b 函数名

b 文件名：代码行号

b 文件名：函数名

b 代码行号 if 条件语句

info break ——查看断点信息（i b）

delete break 断点编号 ——删除断点（d b 断点编号）

d b ——删除所有断点

watch 变量名——设置观察点，变量值变化时停止程序执行命令

c 或者 continue ——继续执行

s 或者 step ——单步执行（进入函数）

n 或者 next ——单步执行（不进入函数）

finish 或者 f ——结束当前函数

变量运行时数据 print 或者 p

p 变量

p 表达式

回溯命令

backtrace<n> 或者 bt<n>——打印栈顶 n 层的信息

make 和 Makefile

项目管理工具， 自动完成编译工作 ，避免不必要的重复编译， Make 通过 makefile 文件来完成并自动维护编译工作。

makefile

makefile 文件中定义了源文件之间的依赖关系
定义如何编译各个源文件并连接生成可执行文件
make 或 make －f makefilename

myapp 的例子如下：

```
#This is a example for describing makefile
myapp:server.o client.o
<TAB>gcc server.o client.o –o myapp
server.o:server.c
<TAB>gcc –c server.c
client.o:client.c
<TAB>gcc –c client.c
clean:
<TAB>rm –f *.o
```

1.3 实验仪器（硬件设备与软件）

(1) PC 机（已安装 Windows 操作系统）；
(2) Linux 操作系统软件（RedHat）或者 VMware 已安装的系统文件包；
(3) VMware 软件和 SSH 客户端软件（SSH Secure Shell）。

1.4 实验内容与步骤

1. 在宿主机器（windows）平台安装 VMware 软件和 SecureShell 客户端软件；
2. 在 VMware 软件上搭建 Linux 客户机平台（采用直接拷贝已安装系统或者系统安装）；
3. VMware 软件网络配置并配置客户机网络，启动虚拟机机 SSH 服务；
4. 并通过宿主机的 SSH 客户端连接虚拟机，实现宿主机与客户机间的文件传输；
5. 学习与验证上文提到的常用命令具体功能。
6. 设计一个程序，达到以下目的：

（1） 使用**变元表**，打印出变元表个数、具体变元表数据；
（2） 获取环境变量（程序参数、外部变量、库函数中选择一种方式实现）；
（3） 通过系统调用获取进程 ID 和父进程 ID，获取进程的用户 ID、有效用户 ID；
（4） 使用 GCC 编译器编译上述程序，并运行程序。
（5） 尝试使用 make 与 makefile 进行项目管理
（6） 使用 GDB 调试程序

1.5 实验报告

实验报告要求：每位同学必须严格安装学校实验报告的要求书写实验报告；涉及到关键命令输出内容、编程代码及其运行输出内容，可打印后附在实验报告中。

1.记录安装配置软件的主要过程与注意事项，特别是实践中遇到的问题与解决方法
2.记录常用命令的验证，及选项功能验证，输出内容
3.记录 GDB 调试过程中命令的使用与输出内容
4.主要代码与关键输出

5.总结实验心得

6.. 讨论当前阶段我们可以在该平台上开展哪些研究工作。

1.6 参考资料

http://www.vmware.cn 等

1.7 准备软件

Secure Shell Client 软件

2 实验二 TCP 协议的理解及套接口编程

2.1 实验目的

（1） 理解 TCP 协议：
- i. 掌握 TCP 协议的报文格式
- ii. 掌握 TCP 连接的建立和释放过程
- iii. 掌握 TCP 数据传输中编号和确认的过程
- iv. 掌握 TCP 协议校验和的计算方法
- v. 理解 TCP 重传机制

（2） 掌握 TCP 套接口基本编程：
- i. 掌握套接口地址及其使用
- ii. 掌握 TCP 套接口编程主要函数的使用及其调用流程

2.2 实验原理

2.2.1 TCP 报文格式

<table>
<tr><td colspan="3">源端口</td><td>目的端口</td></tr>
<tr><td colspan="4">序列号</td></tr>
<tr><td colspan="4">确认号</td></tr>
<tr><td>HLEN</td><td>保留</td><td>控制码</td><td>窗口</td></tr>
<tr><td colspan="3">校验和</td><td>紧急指针</td></tr>
<tr><td colspan="3">选项</td><td>填充字节</td></tr>
<tr><td colspan="4">数据</td></tr>
<tr><td colspan="4">……</td></tr>
</table>

6 种不同控制码或标志位，可以在同一时间设置一位或多位。

URG：紧急指针字段值有效

ACK：确认字段值有效

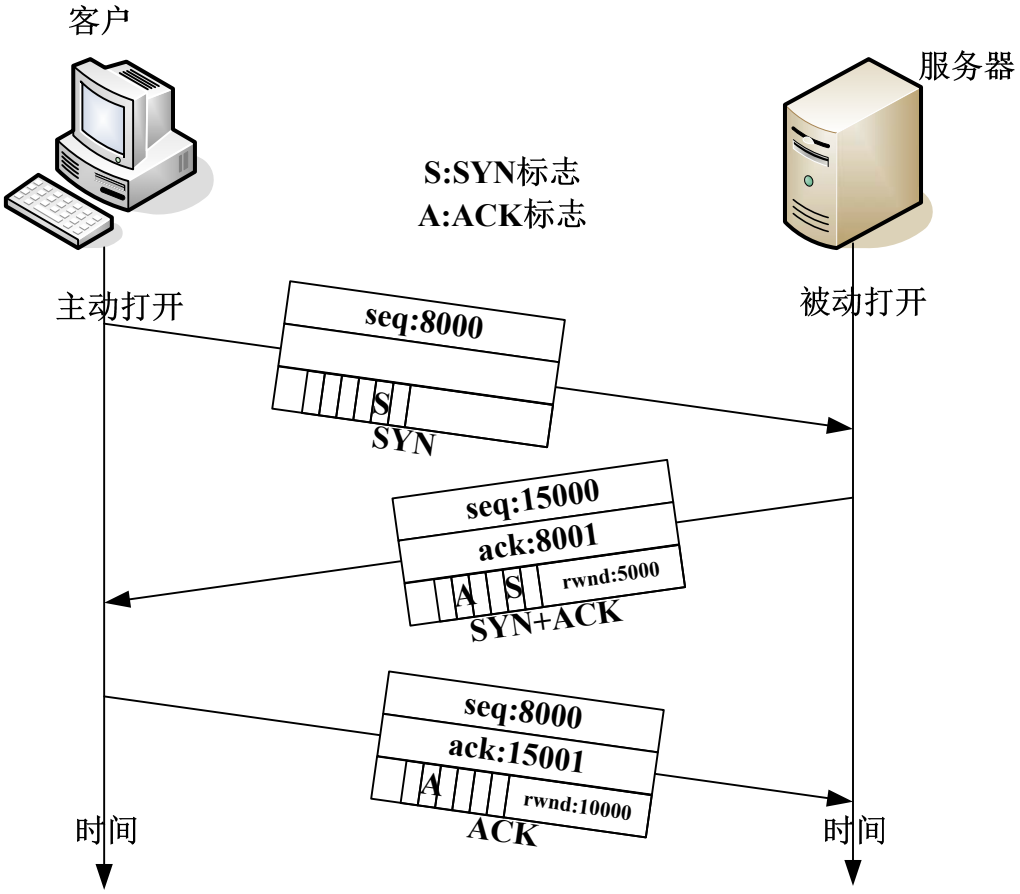
PSH：推送数据

RST：连接必须复位

SYN：建立连接是序号进行同步

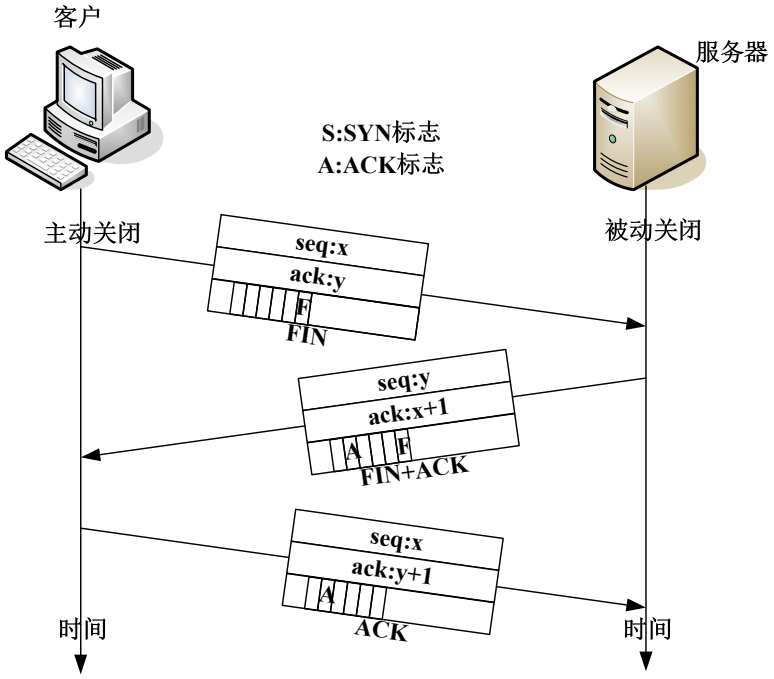
FIN：终止连接

2.2.2 TCP 连接的建立



S：SYN 标志：同步请求；A：ACK 标志：响应请求，ack 为响应的数据编号；seq 为数据编号；rwnd：协商可用缓存窗口大小。

2.2.3 TCP 连接的释放



使用四次握手来结束通话，每一端都要发送一个 FIN 并接收一个 ACK。

2.2.4 TCP 重传机制

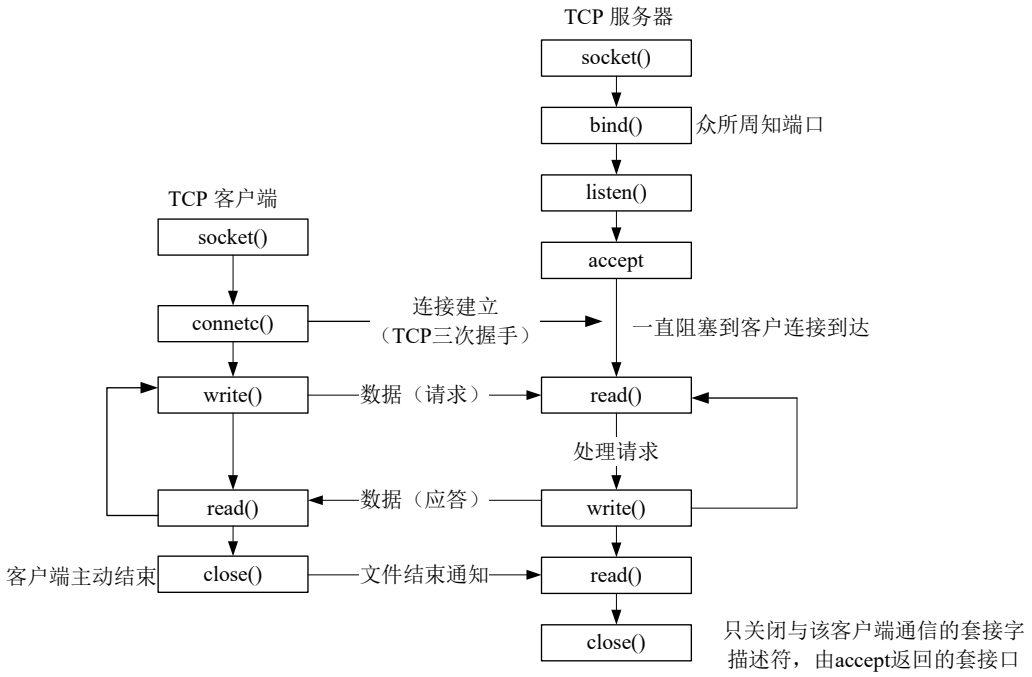
TCP 每发送一个报文段，就对报文段设置一次计时器。只要计时器设置的重传时间到期，还没有收到确认，则重传这一报文段。

2.2.5 套接口地址

```
<linux/sock.h>
struct sockaddr
{
    unsigned short sa_family;
    /* 地址类型，AF_xxx，2 个字节 */
    char        sa_data[14];
    /* 协议地址，14 个字节 */
    }；//16 字节

<linux/in.h>
struct in_addr
{
    _u32 s_addr; /*uint 类型 32 位 IPV4 地址，网络字节顺序*/
};
struct sockaddr_in
{
    short int       sin_family;
    /* 地址类型：AF_INET */
    unsigned short int sin_port;
```

```
    /* 端口号,16 位 TCP/UDP 端口号网络字节顺序 */
    struct in_addr      sin_addr;
    /* ?Internet 地址 32 位地址 */
    unsigned char sin_zero[8];/*8 字节未用*/
};//16 字节

#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int inet_aton(const char *cp，struct in_addr *inp);
//点分十进制字符串?网络字节顺序二进制值
unsigned long int inet_addr(const char *cp);
//点分十进制字符串?网络字节顺序二进制值
//以 255.255.255.255 表示出错，不能表示此广播地址
char * inet_ntoa(struct in_addr in);
//网络字节顺序二进制值?点分十进制字符串

#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
int inet_pton(int af, const char *src, void *dst);
//成功返回 1；输入无效表达式格式返回 0；出错返回-1
const char *inet_ntop(int af, const void *src, char *dst, size_t cnt);
//成功返回结果指针 dst；出错返回 NULL
```

2.2.6 TCP 套接口编程主要函数的使用及其调用流程




```
int socket(int domain, int type, int protocol);
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);
```

```
int listen(int s, int backlog);
int connect(int  sockfd,  const  struct sockaddr *serv_addr, socklen_t  addrlen);
int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int close(int fd);
char *fgets(char *s, int size, FILE *stream);
int fputs(const char *s, FILE *stream);
int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
        struct timeval *timeout);
```

2.2.7 winpcap 和 ethereal 软件简要说明

先安装 winpcap 作为 ethereal 软件嗅探的抓包底层调用函数等。

启动 ethereal 以后，选择菜单 Capature->Start 开始嗅探。按一下 stop， 抓的包就会显示在面板中，并且已经完成分析。

capture 选项

interface: 指定在哪个接口（网卡）上抓包。

Limit each packet: 限制每个包的大小，缺省情况不限制

Capture packets in promiscuous mode: 是否打开混杂模式。如果打开，抓取所有的数据包。一般情况下只需要监听本机收到或者发出的包，因此应该关闭这个选项。

Filter：过滤器。只抓取满足过滤规则的包

File：如果需要将抓到的包写到文件中，在这里输入文件名称。

use ring buffer： 是否使用循环缓冲。缺省情况下不使用，即一直抓包。注意，循环缓冲只有在写文件的时候才有效。如果使用了循环缓冲，还需要设置文件的数目，文件多大时回卷。

其他的项可选择缺省

ethereal 的抓包过滤器：（详见软件中帮助与提示）

抓包过滤器用来抓取感兴趣的包，用在抓包过程中。 抓包过滤器使用的是 libcap 过滤器语言，基本结构是： [not] primitive [and|or [not] primitive ...]。

1、在抓包的时候，就先定义好抓包过滤器，这样结果就是只抓到你设定好的那些类型的数据包；

2、把本机收到或者发出的包一股脑的抓下来，然后使用显示过滤器，只让 Ethereal 显示那些你想要的那些类型的数据包；

点击 Capture Filter 可以得到系统提供的例子，可直接适当修改。

etheral 的显示过滤器：（详见软件中帮助与提示）

显示过滤器可以用来找到你感兴趣的包。可以根据 **协议、是否存在某个域、域值、域值之间的比较**来查找你感兴趣的包。

点击 Filter 可以得到系统提供的例子，可直接适当修改。

参考操作流程：

Capture——>interfaces：选择需要抓包的网络接口（可根据 IP 地址），点击相应行的 Prepare，修改 Capture Filter 值（可针对协议和端口号），Start 即可获得该协议和端口上的数据流的信息。

2.3 实验仪器（硬件设备与软件）

(1) PC 机（已安装 Windows 操作系统）；
(2) Linux 操作系统软件（RedHat）或者 VMware 已安装的系统文件包；
(3) VMware 软件和 SSH 客户端软件（SSH Secure Shell）。
(4) winpcap 和 ethereal 软件（可自行下载和安装）

2.4 实验内容与步骤

2.4.1 编写服务端与客户端程序

（1）基本通信编程

客户端要求：

指定客户端的 IP 地址和端口号

与服务端建立 TCP 连接

请求读取文本文件 A（全部小写字母，多行），并将 A 文件发送给服务器端

显示本地和异地协议地址信息

服务端要求：

指定服务器端的端口号，使用通配 IP 地址，监听 TCP 端口

处理客户端的 TCP 连接请求

接受客户端转换请求（将发送的 A 文件接收后转换全部大写字母），并返回给客户端。

显示本地和异地协议地址信息

总结网络编程中的注意事项

（2）多连接编程

客户端：获取两个随机数，分别发送给两台服务器；并扩展到多服务器；多个套接口向多个服务器请求服务。

服务端：两个服务器，1 个将随机数相加，一个将随机数相乘，分别返回给客户端：扩展到多个服务器。

（3）多路复用：

将上述(1)(2)功能实现，改用多路复用 I/O 来实现。客户端（标准输入/读文件，与网络数据到达进行多路复用）每读取一行便发送信息，服务器端（监听套接口与已连接套接口的复用）每次转换一行信息。

2.4.2 采用 netstat 命令，以及 sleep 系统调用，验证 TCP 连接建立的三次握手各种状态和终止的几种状态。

监听状态，已连接状态，半关闭状态，完全关闭后的状态等。

2.4.3 捕获各个 TCP 连接传输数据过程中的各种信息

采用 winpcap 和 ethereal 软件进行捕获，并记录和解释捕获得到的信息（先在 windows 下捕获各类通信协议的数据包如：telnet, http, ssh, ftp 等）。以及获取的数据信息，分析 TCP 数据包的结构与组成。

2.5 实验报告

实验报告要求：每位同学必须严格安装学校实验报告的要求书写实验报告；涉及到关键编程代码及其运行输出内容，可打印后附在实验报告中。

1. 记录试验过程中的输出数据；
2. 试验中的主要代码及其输出。

2.6 准备软件

winpcap 和 ethereal 软件

3 实验三　UDP 协议 ICMP 协议的理解及 UDP 套接口和原始套接口编程

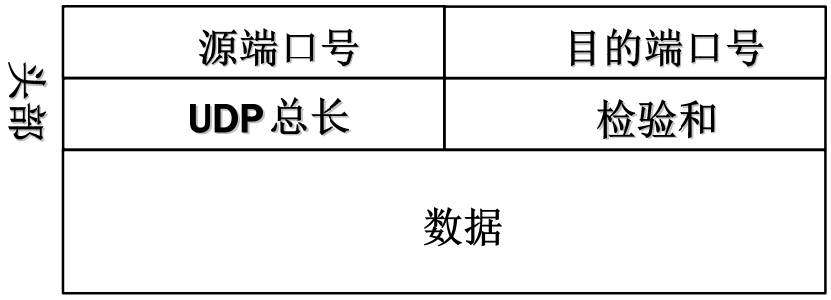
3.1 实验目的

（1） 掌握 UDP 协议的报文格式、理解协议的优缺点

（2） 掌握 ICMP 协议的报文格式、理解不同类型 ICMP 报文的具体意义、了解常见的网络故障

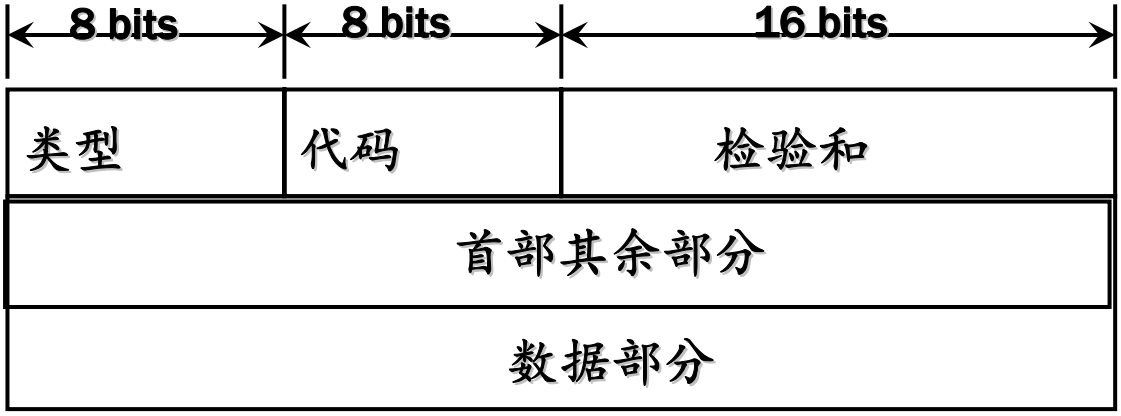
3.2 实验原理

3.2.1 UDP 报文格式

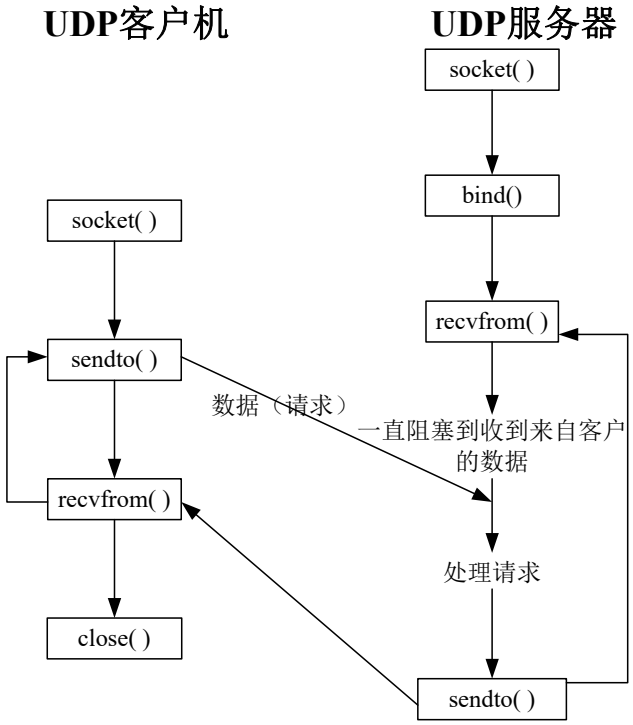


3.2.2 ICMP 报文

种 类	类型	报文	缘由
差错报告报文	3	终点不可达	不可达
	4	源点抑制	拥塞
	11	超时	路由路径过长
	12	参数问题	格式错误
	5	改变路由	路由发生变化
查询报文	8 or 0	回送请求或回答	通路检查
	13 or 14	时间戳请求或回答	实现同步
	17 or 18	地址掩码请求或回答	维护掩码
	10 or 9	路由器询问与通告	路由信息确保一致



3.2.3 UDP 套接口编程主要函数的使用及调用流程



3.3 实验仪器（硬件设备与软件）

(1) PC 机（已安装 Windows 操作系统）；
(2) Linux 操作系统软件（RedHat）或者 VMware 已安装的系统文件包；
(3) VMware 软件和 SSH 客户端软件（SSH Secure Shell）。
(4) winpcap 和 ethereal 软件

3.4 实验内容与步骤

3.4.1 使用 UDP 编写服务器端和客户端程序

（1） 要求：客户端将文件 A1 和 A2，内容交替发送给服务器；服务器大小写转换后交替传给客户端；客户端接收后存为 B1 和 B2 。

（2） 要求：采用连接 UDP 套接口方式，实现（1）功能。启动 2 个服务端，在一个客户端上分别往两个服务器发送 A1 和 A2 文件。

（3） 采用复杂 UDP 并发服务器完成一项编程工作。

3.4.2 使用原始套接口编写发送 ICMP 回射请求，并接收和分析回射应答数据

构建原始套接口，向目标机发送若干个 ICMP 回射请求，接收回射应答 ICMP 数据报，分析后输出到标准输出或指定文件。

3.4.3 上述程序执行时捕获各种数据报的信息

采用 winpcap 和 ethereal 软件进行捕获数据（客户端和服务端在两台 PC 上运行），并记录和解释捕获得到的信息。

3.5 实验报告

实验报告要求：每位同学必须严格安装学校实验报告的要求书写实验报告，全文必须手写，不得打印；涉及到关键编程代码及其运行输出内容，可打印后附在实验报告中。

1. 记录试验过程中的输出数据；
2. 试验中的主要代码及其输出。

4 实验四　带外数据的编程实现

4.1 实验目的

理解带外数据基本原理。

掌握带外数据发送和接收方式。

4.2 实验原理

4.2.1 带外数据的发送方法

在 send 函数中使用 MSG_OOB 标记，实现带外数据的发送。使用标记发送多个字节数据时，最后一个字母为带外数据。

4.2.2 带外数据的接收

【1】 使用 SIGURG 信号，在信号处理函数中调用 recv 函数，并使用 MSG_OOB 标记，接收不在线的带外数据

【2】 采用 select 函数等待套接口描述字的异常条件到达，在 recv 函数中使用 MSG_OOB 标记，接收不在线的带外数据。

【3】 使用 sockatmark 检查带外数据是否到达，再利用 recv 函数接收在线或不在线的带外数据。

4.3 实验仪器（硬件设备与软件）

(1) PC 机（已安装 Windows 操作系统）；

(2) Linux 操作系统软件（RedHat）或者 VMware 已安装的系统文件包；

(3) VMware 软件和 SSH 客户端软件（SSH Secure Shell）。

4.4 实验内容与步骤

验证带外数据的发送与接收的各种方式。

利用带外数据原理设计并实现**客户-服务器心搏函数**。用于发现对端主机或到对端的通信路径的过早失效。

假设每 1 秒钟轮询一次，若持续 5 秒钟没有听到对端应答则认为对端已不再存活，这些值可以有应用程序改动。

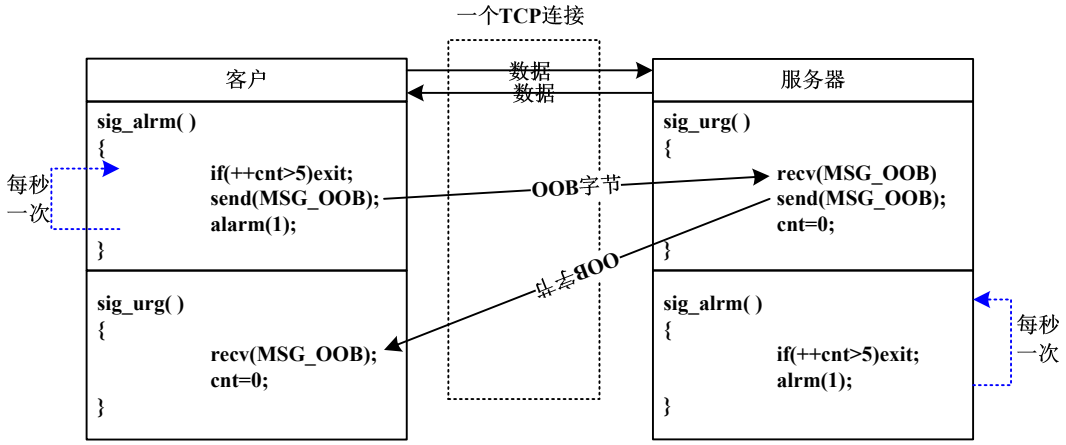
4.4.1 客户-服务器心搏机制

每隔 1 秒钟向服务器发送一个带外字节，服务器收取该字节将导致它向客户发送回一个带外字节。

客户和服务器每秒中递增他们的 cnt 变量一次，每收到一个带外字节又把该变量重置为 0。

若计数器到达 5（即本进程已 5 秒钟没有收到对端的带外字节），就认定连接失效。

双向数据和带外字节都是通过一个 TCP 连接交换。



4.4.2 具体目标

根据上述提示分别设计并实现客户程序心搏函数和服务端心搏函数。并在建立连接的 TCP 两端运行两个心搏函数。

4.4.3 编程提示

客户服务端主函数均在建立连接后运行各自心搏函数，再执行无限循环并在循环中执行 pause 等待。

（客户端心搏函数）

主函数给心搏函数提供：已连接描述字，间隔发送 OOB 的秒数（频率），最大间隔次数数（服务无响应的最大次数）。

建立 SIGURG 和 SIGALRM 信号处理函数，并设置套接口属主为本进程 ID。

按照上图提示编写两个信号处理函数。并在信号处理函数中输出信息到标准输出，以显示心搏函数的正常。或通过人为发送 SIGUSR 信号来终止进程，以测试心搏函数。。

（服务器端心搏函数）

主函数给心搏函数提供：已连接描述字，间隔发送 OOB 的秒数（频率），最大间隔次数数（服务无响应的最大次数）。

建立 SIGURG 和 SIGALRM 信号处理函数，并设置套接口属主为本进程 ID。

按照上图提示编写两个信号处理函数。

4.5 实验报告

实验报告要求：每位同学必须严格安装学校实验报告的要求书写实验报告，全文必须手写，不得打印；涉及到关键编程代码及其运行输出内容，可打印后附在实验报告中。

1. 记录试验过程中的输出数据；
2. 试验中的主要代码及其输出。

5 实验五　阻塞式/非阻塞式 IO

5.1 实验目的

理解阻塞式 IO 与非阻塞 IO 的基本原理和区别
掌握非阻塞式 IO 的编程方法

5.2 实验原理

5.2.1 非阻塞式 IO 的两种设置方法

(1) 函数 fcntl( )，设置 O_NONBLOCK 选项
int flag=fcntl(sockfd,F_GETFL,0);检查文件标志位
fcntl(sockfd,F_SETFL,flag|O_NONBLOCK);设置文件标志位
(2) 函数 ioctl( )，设置 FIONBIO 选项
int nIO=1;设置非阻塞 IO
ioctl(sockfd,FIONBIO,&nIO);

5.3 实验仪器（硬件设备与软件）

(1) PC 机（已安装 Windows 操作系统）；
(2) Linux 操作系统软件（RedHat）或者 VMware 已安装的系统文件包；
(3) VMware 软件和 SSH 客户端软件（SSH Secure Shell）。

5.4 实验内容与步骤

完成 y=a+b
a 由客户端从标准输入获得，消耗时间有用户自行掌握。
b 由服务端随机生成，消耗时间 t 由服务端的随机确定（不超过 1 分钟）。
要求：

(1) 客户端使用非阻塞式 IO 进行接收数据。
(2) 客户端向服务器发出请求数据 b
(3) 客户端从标准输入获取一个整型
(4) 客户端从服务端获得随机数。
(5) 客户端计算并显示计算结果。
(6) 服务端接到请求后，产生随机消耗时间 t。
(7) 服务端 sleep(t)后返回一个随机数给客户端。
(8) 可参考以下流程图进行实现。

5.5 实验报告

实验报告要求：每位同学必须严格安装学校实验报告的要求书写实验报告，全文必须手写，不得打印；涉及到关键编程代码及其运行输出内容，可打印后附在实验报告中。

1. 记录试验过程中的输出数据；
2. 试验中的主要代码及其输出。